Universidad de Las Palmas de Gran Canaria
Instituto Universitaro de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería

Máster Universitario

Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería

TRABAJO FINAL DE MÁSTER

Simulación URANS 2D del flujo alrededor de un cilindro para $10^4 \leq Re \leq 10^6$ con tratamiento de paredes.

Autor:
D. Miguel Ángel Moreno Díaz

Tutor:
D. Luis Alberto Padrón Hernández

4 de diciembre de 2015

Índice general

1. Introducción **8**

1.1. Motivación 8

1.2. Antecedentes 9

1.3. Objetivos 10

2. Conceptos básicos de dinámica de fluidos **11**

2.1. Introducción 11

2.2. Ecuación de conservación de la masa 11

2.3. Ecuación general de conservación 12

2.4. Flujo turbulento 14

2.4.1. Introducción 14

2.4.2. Modelo RANS turbulento $k-\epsilon$ y $k\,\omega$−SST 14

2.4.2.1. La ecuación de Reynolds 14

2.4.2.2. Hipótesis Boussinesq 16

2.4.2.3. La cascada de energía 17

2.4.2.4. Ecuación de la energía turbulenta cinética 17

2.4.2.5. Aproximaciones 19

2.4.2.6. El modelo $k-\epsilon$ 20

2.4.2.7. Modelo $k-\omega$ de Wilcox 21

2.4.2.8. Modelo $k\,\omega$−SST de Menter 23

2.4.2.9. El problema de las dos dimensiones 25

2.5. Capa Límite 25

2.5.1. Capa límite laminar 25

2.5.2. Capa límite turbulenta 26

3. Metodología **27**

3.1. El método de los volúmenes finitos 27

3.1.1. Discretización numérica por el método de volúmenes finitos 28

3.1.1.1. Definiciones generales de la metodología numérica 28
3.1.1.2. Fundamentos del método de volúmenes finitos 30
3.2. Herramienta numérica OpenFOAM 33
3.2.1. Estructura de archivos del estudio 33
3.2.2. Ejecución de una simulación 35
3.2.3. Ejecución de una simulación en paralelo 35
3.2.4. Descomposición de la malla y campos iniciales 36
3.2.5. Ejecución de un caso descompuesto 37
3.2.6. Post-procesado de un caso descompuesto 37
3.3. Funciones de muro (wall function) 38
3.3.1. Funciones de muro estándar 38
3.3.2. Formulación de la WFBC standard 42
3.3.3. Implementación de la Wall Function en OpenFOAM 45
3.3.4. Cálculo de y^+ en OpenFOAM 46
3.3.5. Tratamiento automático de paredes 48
3.3.6. Cálculo del tratamiento automático en paredes en OpenFOAM . . . 48
3.3.7. Código de la función de pared estándar en OpenFOAM 50

4. Resultados **51**
4.1. Definición del problema 51
4.2. Características del dominio computacional y de la discretización 52
4.2.1. Mallados 52
4.2.2. Condiciones de contorno e iniciales 54
4.2.3. Presiones 54
4.2.4. Velocidades 54
4.2.5. k 55
4.2.6. epsilon 55
4.2.7. omega 55
4.2.8. viscosidad ν 56
4.2.9. viscosidad $\nu\prime$ 56
4.3. Definición de las magnitudes de estudio 57
4.3.1. Coeficiente de resistencia 58
4.3.2. Coeficiente de sustentación 58
4.3.3. Número de Strouhal 58
4.4. Parámetros que caracterizan el efecto de los flujos transversales 59
4.4.1. Coeficiente de resistencia 59

4.4.2. Coeficiente de sustentación 60
4.4.3. Número de Strouhal 61
4.4.4. Valores del y^+ 62
4.4.5. Patrones de flujo 63
4.4.5.1. Campos de velocidades obtenidos 63
4.4.5.2. Campos de presiones obtenidos 67
4.5. Estimación de la velocidad de fricción para el diseño de mallas 71

5. Conclusiones y desarrollos futuros **74**

Referencias **76**

Apéndices **78**
5.1. Fichero de datos para el caso $k-\epsilon$ a $Re = 10^4$ 78
5.1.1. Capeta 0 78
5.1.1.1. Archivo epsilon 78
5.1.1.2. Archivo k 79
5.1.1.3. Archivo nut 80
5.1.1.4. Archivo nuTilda 82
5.1.1.5. Archivo p 83
5.1.1.6. Archivo U 84
5.1.2. Carpeta constant 85
5.1.2.1. Archivo RASProperties 85
5.1.2.2. Archivo transportProperties 85
5.1.2.3. Archivo turbulenceProperties 86
5.1.3. Carpeta polyMesh 86
5.1.3.1. Archivo blockMeshDict.m4 86
5.1.4. Carpeta system 94
5.1.4.1. Archivo controlDict 94
5.1.4.2. Archivo decomposePartDict 95
5.1.4.3. Archivo fvSchemes 96
5.1.4.4. Archivo fvSolution 97
5.2. Fichero de datos para el caso $k-\epsilon$ a $Re = 5 \cdot 10^4$ 100
5.2.1. Capeta 0 100
5.2.1.1. Archivo epsilon 100
5.2.1.2. Archivo k 102
5.2.1.3. Archivo nut 103

5.2.1.4. Archivo nuTilda . . . 104
5.2.1.5. Archivo p . . . 105
5.2.1.6. Archivo U . . . 106
5.2.2. Carpeta constant . . . 107
5.2.2.1. Archivo RASProperties . . . 107
5.2.2.2. Archivo transportProperties . . . 108
5.2.2.3. Archivo turbulenceProperties . . . 108
5.2.3. Carpeta polyMesh . . . 109
5.2.3.1. Archivo blockMeshDict.m4 . . . 109
5.2.4. Carpeta system . . . 116
5.2.4.1. Archivo controlDict . . . 116
5.2.4.2. Archivo decomposePartDict . . . 118
5.2.4.3. Archivo fvSchemes . . . 119
5.2.4.4. Archivo fvSolution . . . 120
5.3. Fichero de datos para el caso $k-\epsilon$ a $Re=10^5$ . . . 122
5.3.1. Capeta 0 . . . 122
5.3.1.1. Archivo epsilon . . . 122
5.3.1.2. Archivo k . . . 123
5.3.1.3. Archivo nut . . . 124
5.3.1.4. Archivo nuTilda . . . 125
5.3.1.5. Archivo p . . . 126
5.3.1.6. Archivo U . . . 127
5.3.2. Carpeta constant . . . 128
5.3.2.1. Archivo RASProperties . . . 128
5.3.2.2. Archivo transportProperties . . . 129
5.3.2.3. Archivo turbulenceProperties . . . 129
5.3.3. Carpeta polyMesh . . . 130
5.3.3.1. Archivo blockMeshDict.m4 . . . 130
5.3.4. Carpeta system . . . 138
5.3.4.1. Archivo controlDict . . . 138
5.3.4.2. Archivo decomposePartDict . . . 139
5.3.4.3. Archivo fvSchemes . . . 140
5.3.4.4. Archivo fvSolution . . . 141
5.4. Fichero de datos para el caso $k\,\omega-$SST a $Re=10^4$ . . . 143
5.4.1. Capeta 0 . . . 143
5.4.1.1. Archivo epsilon . . . 143

5.4.1.2. Archivo k . . . 144
5.4.1.3. Archivo nut . . . 145
5.4.1.4. Archivo nuTilda . . . 146
5.4.1.5. Archivo omega . . . 147
5.4.1.6. Archivo p . . . 149
5.4.1.7. Archivo U . . . 150
5.4.2. Carpeta constant . . . 151
5.4.2.1. Archivo RASProperties . . . 151
5.4.2.2. Archivo transportProperties . . . 151
5.4.2.3. Archivo turbulenceProperties . . . 152
5.4.3. Carpeta polyMesh . . . 152
5.4.3.1. Archivo blockMeshDict.m4 . . . 152
5.4.4. Carpeta system . . . 160
5.4.4.1. Archivo controlDict . . . 160
5.4.4.2. Archivo decomposePartDict . . . 161
5.4.4.3. Archivo fvSchemes . . . 162
5.4.4.4. Archivo fvSolution . . . 163
5.5. Fichero de datos para el caso $k-\epsilon$ a $Re = 5 \cdot 10^4$ . . . 165
5.5.1. Capeta 0 . . . 165
5.5.1.1. Archivo epsilon . . . 165
5.5.1.2. Archivo k . . . 166
5.5.1.3. Archivo nut . . . 167
5.5.1.4. Archivo nuTilda . . . 169
5.5.1.5. Archivo omega . . . 170
5.5.1.6. Archivo p . . . 171
5.5.1.7. Archivo U . . . 172
5.5.2. Carpeta constant . . . 173
5.5.2.1. Archivo RASProperties . . . 173
5.5.2.2. Archivo transportProperties . . . 173
5.5.2.3. Archivo turbulenceProperties . . . 174
5.5.3. Carpeta polyMesh . . . 174
5.5.3.1. Archivo blockMeshDict.m4 . . . 174
5.5.4. Carpeta system . . . 182
5.5.4.1. Archivo controlDict . . . 182
5.5.4.2. Archivo decomposePartDict . . . 184
5.5.4.3. Archivo fvSchemes . . . 184

5.5.4.4. Archivo fvSolution 186
5.6. Fichero de datos para el caso $k-\epsilon$ a $Re=10^5$ 187
5.6.1. Capeta 0 187
5.6.1.1. Archivo epsilon 187
5.6.1.2. Archivo k 189
5.6.1.3. Archivo nut 190
5.6.1.4. Archivo nuTilda 191
5.6.1.5. Archivo omega 193
5.6.1.6. Archivo p 194
5.6.1.7. Archivo U 195
5.6.2. Carpeta constant 196
5.6.2.1. Archivo RASProperties 196
5.6.2.2. Archivo transportProperties 197
5.6.2.3. Archivo turbulenceProperties 197
5.6.3. Carpeta polyMesh 198
5.6.3.1. Archivo blockMeshDict.m4 198
5.6.4. Carpeta system 206
5.6.4.1. Archivo controlDict 206
5.6.4.2. Archivo decomposePartDict 208
5.6.4.3. Archivo fvSchemes 209
5.6.4.4. Archivo fvSolution 211
5.7. Fichero de datos para el caso $k-\epsilon$ a $Re=10^6$ 212
5.7.1. Capeta 0 212
5.7.1.1. Archivo epsilon 212
5.7.1.2. Archivo k 214
5.7.1.3. Archivo nut 215
5.7.1.4. Archivo nuTilda 216
5.7.1.5. Archivo omega 217
5.7.1.6. Archivo p 218
5.7.1.7. Archivo U 219
5.7.2. Carpeta constant 220
5.7.2.1. Archivo RASProperties 220
5.7.2.2. Archivo transportProperties 221
5.7.2.3. Archivo turbulenceProperties 221
5.7.3. Carpeta polyMesh 222
5.7.3.1. Archivo blockMeshDict.m4 222

5.7.4. Carpeta system 229
5.7.4.1. Archivo controlDict 229
5.7.4.2. Archivo decomposePartDict 231
5.7.4.3. Archivo fvSchemes 232
5.7.4.4. Archivo fvSolution 233

Capítulo 1

Introducción

1.1. Motivación

La predicción numérica de las vibraciones inducidas por el desprendimiento de vórtices ha sido el foco de numerosas investigaciones hasta la fecha, usando para investigar tal fin herramientas como la dinámica de fluidos computacional. En particular, el flujo alrededor de un cilindro circular ha acaparado mucha atención ya que representa un problema crítico en ingeniería como podemos ver en los cables marinos y en los *drilling risers*. Conocer la frecuencia a la que se desprenden estos vórtices nos permitiría conocer de antemano si ésta coincide con la frecuencia natural de oscilación de la estructura en estudio.

En el presente trabajo se realiza un estudio sobre la adecuación de los modelos de turbulencia $k-\epsilon$ y $k\,\omega-$SST , a diversos números de Reynolds para la predicción de fuerzas sobre un cilindro infinito sumergido en el seno de una masa fluida e incompresible. Los resultados obtenidos serán comparados por los presentados por otros investigadores. Éste trabajo final de máster se encuentra dividido en las siguientes secciones:

- Introducción.
- Conceptos básicos.
- Resultados.
- Conclusiones.

Figura 1.1: Cable submarino

Figura 1.2: Drillling riser

1.2. Antecedentes

Gracias a las mejoras sucesivas en el rendimiento de las computadoras, los métodos numéricos han experimentado una gran acogida, expandiéndose cada vez más y más sus aplicaciones. La aparición de software libre o de código abierto ha permitido a su vez que muchas herramientas informáticas estén al alcance de estudiantes e investigadores sin coste alguno. Entre estos se encuentra el OpenFOAM, software de código abierto para realizar simulaciones fluidodinámicas principalmente, cuyo método numérico de cálculo está basado en el concepto de los volúmenes finitos.

El presente Trabajo fin de Máster se realiza en el marco del Máster Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería impartido por La Universidad de Las Palmas de Gran Canaria, y está integrado en las líneas de trabajo que llevan a cabo los miembros de la División de Mecánica de Medios Continuos y Estructuras en el campo de la Dinámica de Estructuras y, en este caso concreto, en el conocimiento de las características de las cargas dinámicas que actúan sobre las estructuras.

Por otro lado, se ha hecho uso del clúster Leibniz del Centro de Altas Prestaciones del SIANI (CAPSIANI), que está formado por 28 nodos de cálculo y un nodo de acceso o

front-end. Los nodos de cálculo se agrupan en siete equipos *Bullx R424E2*, cada uno con cuatro nodos de cálculo, y fuentes de alimentación redundantes. Cada nodo de cálculo está constituido por:

- 2 procesadores *Intel Xeon E5645 Westmere-EP*, con 6 núcleos cada uno.
- 48 GB de memoria RAM.
- Un disco duro de 500 GB.
- Interfaz *Infiniband* a 40 GBs.

1.3. Objetivos

En el presente Trabajo Final de Máster se realizará un estudio sobre las fuerzas que sufre un cilindro de longitud infinita y rígido sumergido en el seno de un flujo transversal. Para ello se modelizará el problema con dos modelos de turbulencia, $k-\epsilon$ y $k\ \omega-$SST, y se determinará para cada uno de ellos los coeficientes de arrastre y sustentación C_D y C_L a distintos números de Reynolds: $Re = 10^4$, $Re = 5 \cdot 10^4$, $Re = 10^5$ y $Re = 10^6$, este último sólo para el caso $k\ \omega-$SST. Los resultados obtenidos se compararán con resultados empíricos de referencia obtenidos de la literatura científica. De este modo, podrá evaluarse la aptitud del modelo propuesto para el cálculo computacional de las variables estudiadas.

Aunque se puede resolver este problema con una malla lo suficientemente pequeña como para discretizar la subcapa viscosa de la capa límite, es interesante explorar si es posible moderar la subcapa logarítmica de ésta, consiguiendo así un importante ahorro en el número de grados de libertad del problema. Esta modelización se la capa límite para el caso del modelo de turbulencia $k-\epsilon$ es la llamada *Función de Pared Estándar*, y en el caso del modelo de turbulencia $k\ \omega-$SST se ha usado el llamado *Tratamiento Automático de Paredes*. Tanto una modelización de la capa límite como la otra son realmente condiciones de contorno del tipo Dirichlet en la superficie sólida del cilindro, que en el primer caso nos otorgará los valores ϵ en dicha capa, y en el otro los valores de ω.

De esta manera se profundizará, a su vez, en el diseño de mallas para este tipo de problemas en función del número de Reynolds y del valor del parámetro y^+ con el que se desea diseñar la malla. Para tal caso se propondrán expresiones para determinar a priori las velocidades de fricción en este rango, y con ello el tamaño de las celdas necesarias en la malla.

Capítulo 2

Conceptos básicos de dinámica de fluidos

2.1. Introducción

En física, la dinámica de fluidos es una subdisciplina dentro de la mecánica de fluidos. A su vez ésta incluye otras subdisciplinas como son la aerodinámica (estudio del aire y gases en movimiento) y la hidrodinámica (estudio de los fluidos en movimiento). La dinámica de fluidos tiene un amplio abanico de aplicaciones, incluyendo el cálculo de fuerzas y momentos en aviones, cálculo del flujo de petróleo en tuberías, predicción de patrones climáticos, estudio de las nebulosas en el universo e incluso modelar la detonación de armas nucleares. Algunos de sus principios incluso son usados en ingeniería del tráfico donde el tráfico es tratado como un fluido continuo.

La solución de un problema de dinámica de fluidos típicamente envuelve calcular varias propiedades del fluido, tales como velocidad, presión, densidad y temperatura en función del tiempo y del espacio (ver [6]).

2.2. Ecuación de conservación de la masa

La ecuación de conservación de la masa, o ecuación de continuidad, establece de forma general que el incremento de masa en el interior de un elemento fluido es consecuencia del flujo neto de masa hacia dicho elemento. Puesto que en general no puede crearse ni destruirse masa (salvo en casos donde haya involucrada reacciones nucleares), la expresión no estacionaria, tridimensional, en un punto para un fluido general es (ver [6]):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \tag{2.1}$$

En el caso particular de flujo **incompresible** la densidad del fluido no varía temporalmente ni espacialmente en el dominio, por lo que la ecuación se reduce a que la divergencia de la velocidad debe ser nula: $\nabla \cdot \vec{u} = 0$.

2.3. Ecuación general de conservación

Sea ϕ una **variable específica o intensiva**, es decir, cantidades expresadas por unidad de masa, definida sobre un volumen de control de dimensiones $\Delta x \Delta y \Delta z$, la variación temporal de la variable en dicho volumen de control se puede considerar como la suma del flujo neto de dicha variable más la generación de ella en el volumen de control.

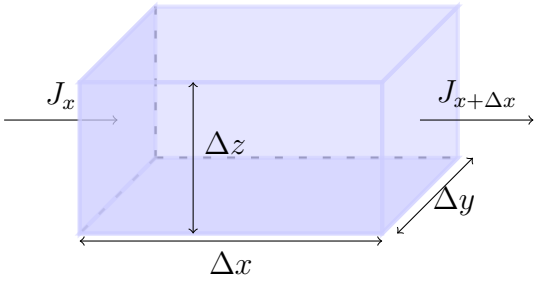


Figura 2.1: Volumen de control.

- El incremento de ϕ en el volumen de control en un paso de tiempo dado es (ver [6])

$$\Delta\phi = (\rho\phi\Delta V)_{t+\Delta t} - (\rho\phi\Delta V)_t \tag{2.2}$$

  donde ΔV es el volumen del elemento de control, t el tiempo y ρ es la densidad.

- La generación neta de ϕ en el volumen de control se puede expresar de manera genérica como $S\ \Delta V\ \Delta t$, donde S representa la generación (*fuente*) de la variable por unidad de volumen. Normalmente se le denomina *término fuente*.

- J es el flujo neto de la variable ϕ por las superficies de control. Ésta se analiza definiendo los flujos perpendiculares a cada superficie del cubo 2.1. Por ejemplo, el flujo neto en la dirección x se define como $(J_x - J_{x+\Delta x})\,\Delta y \Delta z \Delta t$. Así, el flujo neto dentro del volumen de control se puede escribir como

$$(J_x - J_{x+\Delta x})\,\Delta y \Delta z \Delta t + (J_y - J_{y+\Delta y})\,\Delta x \Delta z \Delta t + (J_z - J_{z+\Delta z})\,\Delta y \Delta x \Delta t \tag{2.3}$$

  Los dos mecanismos principales responsables del flujo J son la **convección** y la **difusión**. De esta manera se puede descomponer el flujo J en la suma,

$$J_x = \left(\rho u\phi - \Gamma\frac{d\phi}{dx}\right)_x$$
$$J_{x+\Delta x} = \left(\rho u\phi - \Gamma\frac{d\phi}{dx}\right)_{x+\Delta x} \tag{2.4}$$

donde $(\rho u)_x$ es el flujo másico a través de la cara x, y $\left(\Gamma\frac{d\phi}{dx}\right)$ es la difusión, el cual va acompañada del signo negativo ya que el gradiente de ϕ es negativo, y multiplicada por un coeficiente regulador Γ.

Introduciendo las ecuaciones anteriores en el balance de ϕ en el volumen de control que mencionamos antes, se llega a la expresión de,

$$\overbrace{\frac{\partial\rho\phi}{\partial t}}^{temporal} + \overbrace{\nabla\cdot(\rho\vec{u}\phi)}^{convectivo} = \overbrace{\nabla\cdot(\Gamma\nabla\phi)}^{difusivo} + \overbrace{S}^{fuente} \tag{2.5}$$

Ésta es la ecuación del transporte de la variable ϕ. Para llegar desde ella hasta la ecuación de Navier–Stokes se deben realizar los siguientes cambios de variables:

- $\phi \longrightarrow (u, v, w)$
- $\Gamma \longrightarrow \mu$
- $S \longrightarrow -\overrightarrow{\nabla p} + (S_x, S_y, S_z)$
- $S_y = \rho\vec{g}$ más otras fuentes.

y así, después de realizar unos simplificaciones matemáticas se llega a

$$\rho\frac{\partial\vec{u}}{\partial t} + \rho(\vec{u}\cdot\nabla)\cdot\vec{u} = -\nabla p + \rho\vec{g} + \nabla(\mu\vec{u}) \tag{2.6}$$

que es la expresión general para un fluido **incompresible** y **newtoniano**, es decir, presenta una relación lineal entre las tensiones cortantes y la deformación lineal $\tau = \mu\frac{\partial u}{\partial y}$.

La ecuación 2.6 establece una relación entre las velocidades de cada partícula de fluido (en términos macroscópicos) junto con el término fuente de presiones. En ingeniería normalmente tanto el campo de presiones como el de velocidades son desconocidos sabiéndose de ellos sólo las condiciones de contorno e iniciales. Para completar el sistema de ecuaciones se debe combinar la ecuación 2.6 junto con la ecuación 2.1 mediante los algoritmos de acoplamiento. Una muestra de estos algoritmos son: el Algoritmo SIMPLE *(Semi-Implicit Method for Pressure Linked Equations)*, y sus variantes SIMPLER y SIMPLEC, *(Revisado y Consistente)*, el algoritmo PISO *(Pressure Implicit with Splitting Operators)*, etc (ver [6]).

2.4. Flujo turbulento

2.4.1. Introducción

La turbulencia es el cambio caótico de los campos escalares y vectoriales, presiones y velocidades, en el seno de un fluido. Aunque las ecuaciones de Navier–Stokes describen a la perfección el comportamiento de un fluido, el nivel de mallado requerido para capturar todos los microtorbellinos haría impracticable la utilización de métodos numéricos para resolver Navier–Stokes con los ordenadores de los que disponemos hoy en día. Puesto que estos microtorbelllinos son un medio de disipación de energía, la solución para solventar este inconveniente pasaría por intentar modelar dicha disipación de energía sin tener que llegar a niveles de mallado tan exigentes.

2.4.2. Modelo RANS turbulento $k-\epsilon$ y $k\,\omega-$SST

La base fundamental de la dinámica de fluidos son las ecuaciones de Navier-Stokes. La forma incompresible de estas ecuaciones y la ecuación de continuidad para flujos incompresible son (ver [13]);

$$\frac{Du_i}{Dt} \equiv \frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial u_i}{\partial x_j \partial x_j} \tag{2.7}$$

y

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{2.8}$$

2.4.2.1. La ecuación de Reynolds

En las aproximaciones a la turbulencia RANS (Reynolds Average Navier-Stokes), todas las inestabilidades del flujo son promediadas y consideradas como parte de la turbulencia. Las variables del flujo son representadas como la suma de dos partes, siendo el siguiente ejemplo una de las componentes de la velocidad (ver [13]):

$$u_i(x_i, t) = \overline{u_i}(x_i) + u_i'(x_i, t) \tag{2.9}$$

donde

$$\overline{u_i}(x_i) = \lim_{T\to\infty} \frac{1}{T} \int_0^T u_i(x_i, t) dt \tag{2.10}$$

Aquí T es el intervalo promediado, y debe ser largo comparado con la típica escala de tiempo de fluctuaciones, y u_i' es la fluctuación promediada en el tiempo.

Si el flujo no es estacionario, no se puede promediar en el tiempo y debe ser reemplazado por un *promedio del grupo (ensemble averaging)*. Este último concepto sería imaginar un conjunto de fluidos en el que todas las variables que pueden ser controladas (energía, condiciones de contorno, etc.) son idénticas pero las condiciones iniciales se generan de manera aleatoria. Esto dará flujos que difieren considerablemente unos de otros. Un promedio sobre un gran conjunto de estos flujos es un *ensemble average*. Escrita en forma matemática:

$$\overline{u_i}(x_i,t) = \frac{1}{N}\sum_{n=1}^{N} u_{ni}(x_i,t) \tag{2.11}$$

El término *Reynolds average* aplicado la ecuación de continuidad para fluidos incompresibles,

$$\frac{\partial \overline{u_i}}{\partial x_j} = 0 \tag{2.12}$$

Tomando medias de la parte izquierda de la ecuación 2.7

$$\overline{\frac{Du_i}{Dt}} = \frac{\partial \overline{u_i}}{\partial t} + \overline{\frac{\partial (u_i u_j)}{\partial x_j}} \tag{2.13}$$

Usando la descomposición de 2.9 para el término no linear resulta en:

$$\begin{aligned}
\overline{u_i u_j} &= \overline{(\overline{u_i} + u_i')(\overline{u_j} + u_j')} \\
&= \overline{\overline{u_i}\overline{u_i} + u_j'\overline{u_i} + u_i'\overline{u_j} + \overline{u_i' u_j'}} \\
&= \overline{\overline{u_i}\overline{u_i}} + \overline{u_j'\overline{u_i}} + \overline{u_i'\overline{u_j}} + \overline{u_i' u_j'} \\
&= \overline{u}_i\overline{u}_j + \overline{u_i' u_j'}
\end{aligned} \tag{2.14}$$

ya que

$$\overline{u_j'\overline{u_i}} = \overline{u_i'\overline{u_j}} = 0 \tag{2.15}$$

Si unimos los resultados de las ecuaciones 2.13 y 2.14 obtendremos

$$\overline{\frac{Du_i}{Dt}} \equiv \frac{\partial \overline{u_i}}{\partial t} + \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j} + \overline{u_i}\frac{\partial \overline{u_j}}{\partial x_j} + \frac{\partial \overline{(u_i' u_j')}}{\partial x_j} \tag{2.16}$$

Como el flujo es incompresible, el gradiente $\frac{\partial \overline{u_j}}{\partial x_j}$ es igual a cero, entonces nos queda que

$$\overline{\frac{Du_i}{Dt}} = \frac{\partial \overline{u_i}}{\partial t} + \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{(u_i' u_j')}}{\partial x_j} \tag{2.17}$$

Tomar las medias del resto de términos en la ecuación de momento es simple, ya que la derivada espacial conmuta con el operador de tomar-media.

$$\frac{\partial \overline{u_i}}{\partial t} + \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu\frac{\partial \overline{u_i}}{\partial x_j \partial x_j} - \frac{\partial \overline{(u_i' u_j')}}{\partial x_j} \tag{2.18}$$

La ecuación 2.18 se puede escribir como

$$\rho\left(\frac{\partial \overline{u_i}}{\partial t} + \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j}\right) = \frac{\partial}{\partial x_j}\left[-\overline{p}\delta_{ij} + \mu\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - \rho\overline{u_i' u_j'}\right] \tag{2.19}$$

El término entre corchetes representa la suma de tres tensiones; $\overline{p}\delta_{ij}$ de la media del campo de presiones, las tensiones viscosas de la transferencia de momento a nivel molecular, y el término de tensiones $-\rho\overline{u_i' u_j'}$ que proviene de las fluctuaciones en el campo de velocidades. Este término es el llamado tensiones de Reynolds.

Las tensiones de Reynolds son son las componentes de un tensor simétrico de segundo orden. Las componentes diagonales son tensiones normales y el resto son las tensiones cortantes. La energía cinética turbulenta k es un medio de la traza del tensor de tensiones de Reynolds,

$$k = \frac{1}{2}\rho\overline{u_i' u_j'} \tag{2.20}$$

2.4.2.2. Hipótesis Boussinesq

La hipótesis de la turbulencia viscosa fue introducida por Boussinesq en 1877 y es análoga a relación de las tensiones cortantes para un fluido newtoniano. De acuerdo con esta hipótesis, la relación es (ver [13]):

$$-\overline{u_i' u_j'} = \nu_T\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - \frac{2}{3}k\delta_{ij} \tag{2.21}$$

donde el campo escalar positivo, $\nu_T = \nu_T(x_i, t)$, es la viscosidad turbulenta. Sustituyendo la la viscosidad turbulenta dentro de 2.18 se obtiene:

$$\frac{\partial \overline{u_i}}{\partial t} + \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\nu_{ef}\left(\frac{\overline{u_i}}{\partial x_j} + \frac{\overline{u_j}}{\partial x_i}\right)\right] - \frac{1}{\rho}\frac{\partial}{\partial x_i}\left(\overline{p} + \frac{2}{3}\rho k\right) \tag{2.22}$$

donde

$$\nu_{ef}(x_i, t) = \nu + \nu_T(x_i, t) \tag{2.23}$$

(Viscosidad Efectiva)

La ecuación 2.22 tiene la misma apariencia como la ecuación para el caso incompresible de Navier–Stokes, con $\overline{u_i}$ y ν_{ef} en lugar de u_i y ν, y con $\overline{p} + \frac{2}{3}\rho k$ modificando la presión. La ventaja de este modelo radica en su extrema simpleza. Desafortunadamente, para muchos tipos de flujo su precisión es más bien pobre. Esto viene a demostrar que la física de la turbulencia es diferente de la física de procesos moleculares que conducen a las relaciones de las tensiones viscosas en un fluido newtoniano. Sin embargo, para flujos simples cortantes, donde el gradiente medio de la velocidades y turbulencia característica se desarrollan lentamente, la hipótesis es razonablemente válida.

2.4.2.3. La cascada de energía

El fenómeno de la turbulencia se considera que está compuesto por vórtices de diferentes tamaños. Los vórtices más grandes del flujo son inestables y tienden a romperse, transfiriendo su energía a vórtices más pequeños. Éstos a su vez se van rompiendo y transmitiendo su energía a sucesivos vórtices de menor tamaño. Esta *cascada de energía* continúa hasta que el número de Reynolds $Re \equiv u(l)l/\nu$ es lo suficientemente pequeño, de tal manera que el movimiento y la viscosidad molecular es efectiva disipando la energía cinética.

Esto es importante ya que sitúa la disipación de energía en el último tramo de la cascada de energía. El ratio de disipación denotado como ϵ, está determinado por el primer proceso de la secuencia, el cual es la transferencia de energía de los vórtices más grandes. Estos vórtices están determinados por la longitud de escala l_0, la velocidad de escala u_0, el tiempo de escala $\tau_0 = l_0/u_0$ y posee una energía de $\frac{1}{2}\rho u_0^2$. Por lo que el ratio de transferencia de energía puede ser supuesto como $u_0^2/\tau_0 = u_0^3/l_0$. Consecuentemente, ϵ es u_0^3/l_0, independientemente de la ν (ver [13]).

2.4.2.4. Ecuación de la energía turbulenta cinética

En esta sección se deriva una ecuación diferencial que describe el comportamiento de la energía cinética turbulenta, siguiendo la línea propuesta en [13]. Empezando por multiplicar las ecuaciones incompresibles de Navier–Stokes por u_i y a continuación tomando medias de los campos resultantes.

$$\overline{\frac{\partial u_i}{\partial t} u_i} + \overline{u_j \frac{\partial u_i}{\partial x_j} u_i} = -\frac{1}{\rho}\overline{\frac{\partial p}{\partial x_i} u_i} + \nu \overline{u_i \nabla^2 u_i} \tag{2.24}$$

Multiplicando la ecuación de Reynolds por u_i da,

$$\frac{\partial \overline{u_i}}{\partial t}\overline{u_i} + \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j}u_i = -\frac{\partial \overline{u_i' u_j'}}{\partial x_j}\overline{u_i} - \frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_j}\overline{u_i} + \nu\overline{u_i}\overline{\nabla^2 u_i} \tag{2.25}$$

Teniendo en cuenta las siguientes propiedades de las medias,

$$\overline{u_i u_j} = \overline{u_i u_j} + \overline{u_i' u_j'} \tag{2.26}$$

$$\overline{\frac{\partial u_i}{\partial x_j}} = \frac{\partial \overline{u_i}}{\partial x_j} \tag{2.27}$$

$$\overline{\frac{\partial u_i}{\partial x_j}u_i} = \frac{\partial u_i}{\partial x_j}\overline{u_i} + \overline{\frac{\partial u_i'}{\partial x_j}u_i'} \tag{2.28}$$

$$\overline{u_i u_j u_k} = \overline{u_i' u_j' u_k'} + \overline{u_i' u_j'}\,\overline{u_k} + \overline{u_j' u_k'}\,\overline{u_i} + \overline{u_k' u_i'}\,\overline{u_j} + \overline{u_i u_j u_k} \tag{2.29}$$

Sustrayendo 2.29 de 2.24 obtenemos,

$$\overline{\rho\frac{\partial u_i'}{\partial t}u_i'} + \rho\left(\overline{u_j\frac{\partial u_i}{\partial x_j}u_i} - \overline{u_j}\frac{\partial \overline{u_i}}{\partial x_j}\overline{u_i}\right) = -\overline{\frac{\partial p'}{\partial x_i}u_i'} + \nu\overline{u_i'\nabla^2 u_i'} + \rho\frac{\partial \overline{u_i' u_j'}}{\partial x_j}\overline{u_i} \tag{2.30}$$

De las propiedades de las medias tenemos,

$$\overline{u_j\frac{\partial u_i}{\partial x_j}u_i} - u_j\frac{\partial \overline{u_i}}{\partial x_j}\overline{u_i} = \overline{u_i'\frac{\partial u_i'}{\partial x_j}}\overline{u_j} + \overline{\frac{\partial u_i'}{\partial x_j}u_i' u_j'} + \overline{\frac{\partial u_i'}{\partial x_j}u_j'}\,\overline{u_i'} + \overline{u_i' u_j'}\,\overline{\frac{\partial u_i}{\partial x_j}} \tag{2.31}$$

Usando la ecuación 2.30, la ecuación 2.31, la regla de la cadena de la derivadas, y la incompresibilidad de la media del campo de velocidades, llegamos a que,

$$\begin{aligned}\rho\left(\overline{\frac{\partial u_i'}{\partial t}u_i'} + \overline{u_i'\frac{\partial u_i'}{\partial x_j}}\overline{u_j} + \overline{\frac{\partial u_i'}{\partial x_j}u_i' u_j'} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j}\overline{u_i} + \overline{u_i' u_j'}\,\overline{\frac{\partial u_i}{\partial x_j}}\right) &= -\overline{\frac{\partial p'}{\partial x_i}u_i'} \\ &+\nu\overline{u_i'\nabla^2 u_i'} + \rho\frac{\partial \overline{u_i' u_j'}}{\partial x_j}\overline{u_i}\end{aligned} \tag{2.32}$$

El cuarto término del primer miembro, y el último término del segundo miembro son iguales y se cancelan mutuamente. Usando nuevamente la regla de la cadena tenemos que,

$$\frac{1}{2}\left(\frac{\partial\overline{u_i'u_i'}}{\partial t}+\overline{u_j}\frac{\partial\overline{u_i'u_i'}}{\partial x_j}+\frac{\partial\overline{u_i'u_i'u_j'}}{\partial x_j}\right)=-\overline{u_i'u_j'}\frac{\overline{\partial u_i'u_i'u_j'}}{\partial x_j}-\frac{1}{\rho}\frac{\overline{\partial p'u_i'}}{\partial x_j}$$
$$+\nu\overline{\frac{\partial}{\partial x_j}\left(\frac{1}{2}\frac{\partial u_i'u_i'}{\partial x_j}\right)-\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} \tag{2.33}$$

Usando la definición de energía cinética turbulenta de acuerdo con la ecuación 2.20, y la definición del ratio de disipación de energía turbulenta dada por

$$\epsilon=\nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}} \tag{2.34}$$

se llega finalmente a que,

$$\frac{\partial k}{\partial t}+\overline{u_j}\frac{\partial k}{\partial x_j}=-\frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u_i'u_i'u_j'}+\frac{1}{\rho}\overline{u_j'p'}-\nu\frac{\partial k}{\partial x_j}\right)-\overline{u_j'u_i'}\frac{\partial\overline{u_i}}{\partial x_j}-\epsilon \tag{2.35}$$

La suma de los dos términos en el lado izquierdo, el término no estacionario y el convectivo, es la derivada material sobre k, que da el ratio de cambio de k siguiendo al elemento fluido. El primer término del lado derecho es conocido como el *transporte turbulento*, y es debido al ratio al cual la energía turbulenta es transportada a través del fluido por fluctuaciones turbulentas. El segundo término del lado derecho se conoce como *difusión por la presión*, y es otra forma de transporte turbulento resultado de la correlación de las fluctuaciones de presiones y velocidades. El tercer término en el lado derecho representa la difusión de energía turbulenta causada por procesos naturales a nivel molecular. El cuarto término del lado derecho es conocida como *producción*, y representa el ratio al cual la energía cinética es transportada desde el flujo medio a la turbulencia. Finalmente, ϵ es el ratio de disipación de la energía cinética turbulenta.

2.4.2.5. Aproximaciones

La parte izquierda de la ecuación 2.35, y el término que representa la difusión molecular son exactos, mientras que la producción, la disipación, el transporte turbulento y la difusión por la presión, requieren de correlaciones desconocidas. Para cerrar la ecuación, estos términos deben ser aproximados. La manera estándar de aproximar el transporte turbulento de cantidades escalares es usar la hipótesis de *gradiente-difusión*. En analogía con procesos de transporte molecular, decimos que $-\overline{u_j'\phi'}\approx\mu_T\frac{\partial\phi}{\partial x_j}$, donde ϕ es algún escalar, p o k por ejemplo. Así (ver [13]),

$$\frac{1}{2}\overline{u_i'u_i'u_j'}+\frac{1}{\rho}\overline{u_j'p'}=-\frac{\nu_T}{\sigma_k}\frac{\partial k}{\partial x_j} \tag{2.36}$$

donde σ_k es el *número de Prandtl turbulento* para la energía cinética, el cual es generalmente tomado como la unidad. La hipótesis de *gradiente-difusión* afirma que existe un flujo de k gradiente abajo de k. Matemáticamente, esto asegura un suavizado en las soluciones, y que una condición de contorno se puede imponer sobre k en cualquier contorno. Usando este modelo para el transporte turbulento y la difusión por presión, junto con la ecuación 2.21 para el término de producción, finalizamos en el siguiente modelo del transporte de k:

$$\frac{\partial k}{\partial t} + \overline{u_j}\frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + \left[\nu_T\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - \frac{2}{3}k\delta_{i,j}\right]\frac{\partial \overline{u_i}}{\partial x_j} - \epsilon \quad (2.37)$$

Puesto el campo de las velocidades medias es incompresible,

$$\frac{2}{3}\delta_{i,j}\frac{\partial \overline{u_i}}{\partial x_j} = \frac{2}{3}\frac{\partial \overline{u_j}}{\partial x_j} = 0 \quad (2.38)$$

por lo que la ecuación 2.37 se reduce a

$$\frac{\partial k}{\partial t} + \overline{u_j}\frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + \nu_T\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)\frac{\partial \overline{u_i}}{\partial x_j} - \epsilon \quad (2.39)$$

Puesto que el ratio de disipación escala como u_0^3/l_0, es razonable modelar ϵ como

$$\epsilon = C \cdot k^{3/2}/l(x_i) \quad (2.40)$$

donde $l(x_i)$ es la escala de longitud de la turbulencia, y C es una constante.

En la hipótesis de viscosidad turbulenta, la ecuación 2.21, se introduce la viscosidad turbulenta ν_T. Para cerrar el sistema de ecuaciones, ésta debe ser especificada. Basándose enteramente en argumentos dimensionales, la viscosidad turbulenta está dada por

$$\nu = k^{1/2}l(x_i) \quad (2.41)$$

2.4.2.6. El modelo $k - \epsilon$

Para eliminar la necesidad de especificar la longitud de escala turbulenta $l(x_i)$, en adición a la ecuación de k, una ecuación de transporte para otra cantidad turbulenta más se puede usar. Este tipo de modelos es llamado *modelo de dos ecuaciones* y la estándar es el modelo $k - \epsilon$. En este modelo, una ecuación adicional se resuelve para el transporte de ϵ. La ecuación exacta de ϵ se puede derivar de manera similar a la ecuación de k, pero no es un punto de partida muy útil para la resolución del modelo. Esto es debido a que ϵ se ve mejor como un ratio de flujo de energía turbulenta al comienzo de la cascada de energía, el cual era la transferencia de energía desde los torbellinos más grandes del fluido hacia los más pequeños. En contraste, la ecuación exacta de ϵ pertenece al proceso en rango de disipación, es decir, al final de la cascada. Entonces el modelo estándar para ϵ se ve mejor como si fuese enteramente empírica. La ecuación es [13]

$$\frac{\partial \epsilon}{\partial t}+\overline{u_j}\frac{\partial \epsilon}{\partial x_j}=\frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_k}\frac{\partial \epsilon}{\partial x_j}\right)+C_{\epsilon 1}\frac{\epsilon}{k}\left[\nu_T\left(\frac{\partial \overline{u_i}}{\partial x_j}+\frac{\partial \overline{u_j}}{\partial x_i}\right)-\frac{2}{3}k\delta_{ij}\right]\frac{\partial \overline{u_i}}{\partial x_j}-C_{\epsilon 2}\frac{\epsilon^2}{k} \quad (2.42)$$

Debido a la incompresibilidad del campo medio de velocidades, esta expresión se simplifica a

$$\frac{\partial \epsilon}{\partial t}+\overline{u_j}\frac{\partial \epsilon}{\partial x_j}=\frac{\partial}{\partial x_j}\left(\frac{\nu_T}{\sigma_k}\frac{\partial \epsilon}{\partial x_j}\right)+C_{\epsilon 1}\frac{\epsilon}{k}\nu_T\left(\frac{\partial \overline{u_i}}{\partial x_j}+\frac{\partial \overline{u_j}}{\partial x_i}\right)\frac{\partial \overline{u_i}}{\partial x_j}-C_{\epsilon 2}\frac{\epsilon^2}{k} \quad (2.43)$$

Combinando la ecuación 2.40 y la ecuación 2.41, se puede escribir la viscosidad turbulenta como

$$\nu_T = C_\mu k^2/\epsilon \quad (2.44)$$

y por lo tanto $l(x_i)$ se obtiene de k y de ϵ. Aquí C_μ es una constante del modelo.

La ecuación de k y ϵ junto con la especificación de ν_T, forman el modelo de turbulencia $k-\epsilon$. Se dice que este modelo está completo ya que no requiere ninguna especificación adicional salvo la escala de longitud $l(x_i)$.

El modelo $k-\epsilon$ consta de cuatro componentes; dos modelos de ecuaciones se resuelvan para k y para ϵ. La viscosidad turbulenta se define por $\nu_T = C_\mu k^2/\epsilon$. Las tensiones de Reynolds las encontramos en la hipótesis de la *viscosidad turbulenta*, y se resuelven las ecuaciones de Reynolds para $\overline{u_i}$ y $\overline{p}$.

Los valores estandarizados de las constantes que aparecen en el modelo de turbulencia $k-\epsilon$ son:

$$C_\mu = 0{,}09,\ C_{\epsilon 1} = 1{,}44,\ C_{\epsilon 2} = 1{,}92,\ \sigma_k = 1,\ \sigma_\epsilon = 1{,}3 \quad (2.45)$$

2.4.2.7. Modelo $k-\omega$ de Wilcox

En el modelo $k-\epsilon$, la viscosidad turbulenta cinemática ν_t es expresada como el producto de una velocidad $u=\sqrt{k}$, y una longitud $l = k^{3/2}/\epsilon$. El ratio de disipación de la energía turbulenta cinética ϵ no es la única variable de longitud determinante. De hecho, muchos otros modelos de dos ecuaciones se han postulado. La más prominente es el modelo $k-\omega$ propuesto por Wilcox, el cual usa la frecuencia turbulenta $\omega = \epsilon/k\ [s^{-1}]$ como la segunda variable. Si usamos esta variable, la escala de longitud sería $l = \sqrt{k}/\omega$. La viscosidad turbulenta viene dada por (ver [1])

$$\mu_t = \rho k/\omega \quad (2.46)$$

Las tensiones de Reynolds se computan como se hace en modelos de dos ecuaciones con la expresión de Boussinesq:

$$\tau_{ij} = -\rho\overline{u_i' u_j'} = 2\nu_t S_{ij} - 2\rho k\delta_{ij} = \nu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\rho k\delta_{ij} \tag{2.47}$$

donde S_{ij} es el tensor de deformaciones. La ecuación del transporte para k y para ω, para flujos de Reynolds altos, es como sigue:

$$\frac{(\rho k)}{\partial t} + div(\rho k u) = div\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right) grad(k)\right] + P_k - \beta^* \rho k\omega \tag{2.48}$$

donde

$$P_k = \left(2\mu_t S_{ij} \cdot S_{ij} - \frac{2}{3}\rho k \frac{\partial u_i}{\partial x_j}\delta_{ij}\right) \tag{2.49}$$

es el ratio de producción de energía cinética turbulenta, y

$$\begin{aligned}\frac{\partial(\rho\omega)}{\partial t} + div(\rho\omega u) = div\left[\left(\mu + \frac{\mu_t}{\sigma_\omega}\right)\right] grad(\omega) \\ + \gamma_1 \left(2\rho S_{ij} \cdot S_{ij} - \frac{2}{3}\rho\omega \frac{\partial u_i}{\partial x_j}\delta_{ij}\right) - \beta_1 \rho\omega^2\end{aligned} \tag{2.50}$$

En otras palabras, el ratio de cambio de k y ω, más el transporte de k y ω por convección, es igual al transporte de k y ω por difusión turbulenta, más el ratio de producción de k o ω, menos el ratio de disipación de k o ω.

Las constantes del modelo son:

- $\sigma_k = 2{,}0$
- $\sigma_\omega = 2{,}0$
- $\gamma_1 = 0{,}553$
- $\beta_1 = 0{,}075$
- $\beta^* = 0{,}09$

El modelo $k - \omega$ inicialmente atrajo la atracción porque la integración de las paredes no requiere de *funciones de salto* en aplicaciones para bajos Reynolds. El valor de la energía cinética turbulenta k en las paredes se pone a valor cero. La frecuencia ω tiende a infinito en las paredes, pero podemos especificar un valor muy grande en la pared, o siguiendo a Wilcox, aplicando una variación hiperbólica $\omega = 6\nu/(\beta_1 y_P^2)$ en los puntos adyacentes a paredes. La experiencia ha demostrado que los resultados no dependen mucho en el tratamiento preciso de este detalle.

En los contornos del tipo *inlet* se deben especificar valores de k y ω, y en los contornos de tipo *outlet* usar condiciones del tipo *gradiente zero*. La condición de ω para flujo libre de turbulencias, donde k tiende a cero y la frecuencia ω tiende a cero, es la más problemática. La ecuación 2.46 muestra que la viscosidad turbulenta μ_t es indeterminada o infinita cuando ω tiende a cero, por lo que es necesario darle un valor pequeño pero distinto de cero. Desafortunadamente, los resultados del modelo tienden a ser dependientes del valor de ω asumido en estas zonas libres de turbulencias, lo cual es un serio problema en flujos externos como los que se tienen en el campo de la aerodinámica y aplicaciones aeroespaciales, donde las condiciones de contorno libre son usadas de manera obligatoria.

2.4.2.8. Modelo $k\,\omega$−SST de Menter

Menter notó que los resultados del modelo $k-\epsilon$ son mucho menos sensibles a los valores arbitrarios en las corrientes libres, pero su bondad en el tratamiento de paredes era insatisfactorio para capas límites con gradientes de presiones adversos. Esto le condujo a sugerir un modelo híbrido usando:

1. Una transformación del modelo $k-\epsilon$ en el modelo $k-\omega$ en las regiones cercanas a paredes.
2. El modelo $k-\epsilon$ estándar en las regiones alejadas de las paredes.

El cálculo de las tensiones de Reynolds, y de la ecuación de k, son las mismas que en el modelo $k-\omega$ original de Wilcox, pero la ecuación de ϵ se transforma en una ecuación de ω asumiendo que $\epsilon = k\omega$. Esto conduce a (ver [1])

$$\begin{aligned}\frac{\partial(\rho\omega)}{\partial t} + div(\rho\omega u) = div\left[\left(\mu + \frac{\mu_t}{\sigma_{\omega,1}}\right) grad(\omega)\right] \\ + \gamma_2\left(2\rho S_{ij}\cdot S_{ij} - \frac{2}{3}\rho\omega\frac{\partial u_i}{\partial x_j}\delta_{ij}\right) \\ - \beta_2\rho\omega^2 + 2\frac{\rho}{\sigma_{\omega,2}\omega}\frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k}\end{aligned} \tag{2.51}$$

Comparándola con la ecuación 2.49, se ve que 2.51 tiene un término fuente extra en el lado derecho, el término de difusión cruzada, que surge durante la transformación del término de la difusión en la ecuación de ϵ.

Menter y compañía resumieron una serie de modificaciones para optimizar la bondad del modelo $k\,\omega$−SST basándose en la experiencia con el modelo. Las principales mejoras son:

- Revisión de las constantes del modelo:

  $\sigma_k = 1{,}0$

  $\sigma_{\omega,1} = 2{,}0$

  $\sigma_{\omega,2} = 1{,}17$

$\gamma_2 = 0{,}44$

$\beta_2 = 0{,}083$

$\beta^* = 0{,}09$

- Funciones de mezcla (*Blending Function*). Inestabilidades numéricas pueden ser causadas por existir diferencias en los valores calculados de la viscosidad turbulenta, con el $k-\epsilon$ estándar en las zonas alejadas por las afecciones turbulentas, y el modelo $k-\epsilon$ transformado para el tratamiento de paredes. Las funciones de mezcla se usan para alcanzar transiciones suaves entre ambos modelos. Las funciones de mezcla se introducen en la ecuación para modificar el término de difusión cruzada, y también son introducidas en las constantes del modelo que toman el valor C_1 para el modelo original $k-\omega$, y el valor C_2 en el modelo $k-\epsilon$ transformado de Menter:

$$C = F_C C_1 + (1 - F_C) C_2 \tag{2.52}$$

Típicamente las funciones de mezcla $F_C = F_C(l_t/y,\ Re_y)$ son una función del ratio de turbulencia $l_t = \sqrt{k}/\omega$, la distancia y a la pared, y del Reynolds local $Re = y^2\omega/\mu$. La función de forma F_C es escoge de tal manera que:

Es cero en las paredes.

Tiende a la unidad en el campo alejado de la influencia de turbulencias.

Produzca transiciones suaves a una distancia intermedia entre la pared y la zona donde termina la capa límite.

De esta manera el método combina ahora el buen comportamiento entorno a paredes del modelo $k-\omega$, con la robustez de los modelos $k-\epsilon$ en cuanto a su estabilidad numérica en las zonas libre de afecciones turbulentas.

- Limitadores: La viscosidad turbulenta está limitada a dar rendimientos mejorados en flujos con gradientes de presión adversa y regiones débiles, y la producción de energía cinética turbulenta está limitada para prevenir la aparición de turbulencias en regiones de estancamiento. Los limitadores son:

$$\mu_t = \frac{a_1 \rho k}{max(a_1\omega,\ SF_2)} \tag{2.53}$$

donde $S = \sqrt{2S_{ij} \cdot S_{ij}}$, a_1 = constante, F_2 es una función de mezcla, y

$$P_k = min\left(10\beta^*\rho k\omega,\ 2\mu_t S_{ij} S_{ij} - \frac{2}{3}\rho k \frac{\partial u_i}{\partial x_j}\delta_{ij}\right) \tag{2.54}$$

2.4.2.9. El problema de las dos dimensiones

La predicción numérica sobre las vibraciones inducidas por vórtices ha sido el foco de numerosas investigaciones usando herramientas de dinámica de fluidos computacionales. En particular, el flujo alrededor de un cilindro circular ha acaparado la atención por representar problemas críticos en ingeniería como cables marinos. Las limitaciones debidas al coste computacional impuesta por las soluciones de gran cantidad de ecuaciones ha conducido a que la mayor parte de la literatura especializada estudie solamente flujos bidimensionales con pocas excepciones. Las discrepancias encontradas entre los datos experimentales y los resultados de simulaciones numéricas en 2D sugieren que las inestabilidades que aparecen en 3D aparecen a lo largo del cilindro y que afectan notablemente a las características del fluido. Las escasas simulaciones que se han realizado en 3D confirman esta hipótesis.

2.5. Capa Límite

En física y en mecánica de fluidos, la capa límite es la capa de fluido inmediata a una superficie sólida donde los efectos de la viscosidad son importantes. Es la zona fluida inmediata a una superficie sólida donde las fuerzas viscosas distorsionan el flujo no viscoso adyacente. La naturaleza viscosa del flujo inmediato a las alas reduce el campo de velocidades locales produciendo tensiones cortantes sobre la superficie sólida, *skin friction*.

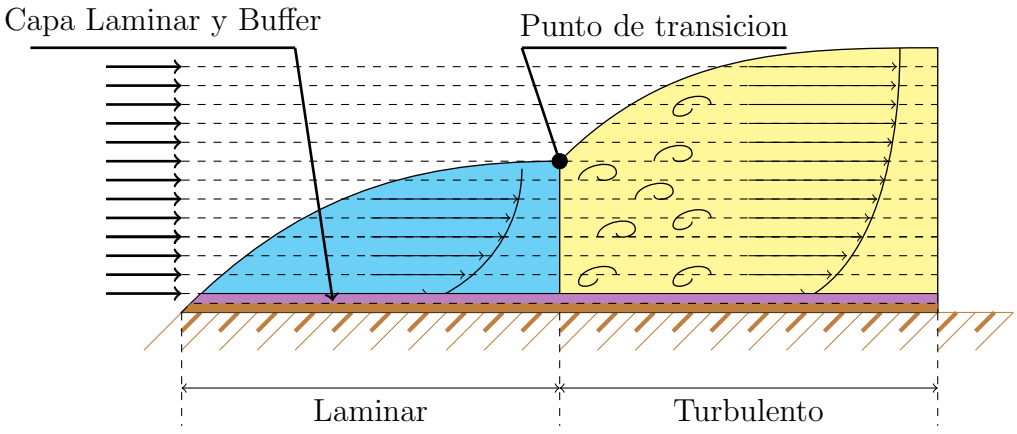


Figura 2.2: Capa Límite.

2.5.1. Capa límite laminar

La capa límite laminar presenta un flujo muy suave, mientras que la capa límite turbulenta contiene vórtices. El flujo laminar crea menos tensión cortante que el flujo turbulento, y por tanto, menor fuerza de arrastre, pero es menos estable. La capa límite sobre una superficie sólida comienza siendo un flujo laminar suave que va aumentando de espesor a medida que éste avanza a lo largo de dicha superficie.

2.5.2. Capa límite turbulenta

A cierta distancia desde que el flujo entra en contacto con la superficie, el flujo laminar se rompe produciéndose una transición hacia un flujo turbulento. Desde un punto de vista de eficiencia aerodinámica, conviene que que esta transición desde el flujo laminar al flujo turbulento se retrase lo máximo posible, aumentando la cantidad de flujo laminar sobre la superficie.

Capítulo 3

Metodología

3.1. El método de los volúmenes finitos

El objeto del método de volúmenes finitos es desarrollar una metodología numérica para resolver la ecuación general de transporte. La idea de partida fundamental es el concepto de discretización: reemplazar una solución analítica en derivadas parciales que proporciona el valor de ϕ de forma continua en todos los puntos del espacio por una solución numérica aproximada que da el valor de ϕ únicamente en una serie de puntos discretos definidos por la malla del dominio.

El método de volúmenes finitos propone una forma de llevar a cabo esa discretización. En particular, establece que los valores discretos de ϕ quedarán descritos por un conjunto de ecuaciones algebraicas que relacionan el valor de la variable en un punto con el valor en los puntos vecinos. La forma en que se transmite la información entre esos nodos requiere de algún tipo de aproximación, que el caso de volúmenes finitos es mediante esquemas conservativos que evalúan los flujos a través de superficies de control.

La transformación de las ecuaciones diferenciales en un conjunto de ecuaciones algebraicas precisa necesariamente de una discretización espacial.

Esto se consigue generando una malla que permite dividir el dominio de interés en una serie de celdas a las cuales se le asocia el valor de la variable discreta ϕ.

La razón por la cual el método de volúmenes finitos es el más empleado para desarrollar códigos CFD reside en su generalidad, su simplicidad conceptual y su facilidad para ser implementado en cualquier tipo de mallado, ya sea estructurado o no estructurado.

Existen una serie de propiedades fundamentales en el método de volúmenes finitos que merece la pena destacar. En primer lugar, hay que recordar que el método discretiza el dominio en un número finito de celdas (o volúmenes de control). Por tanto, el método se basa en valores discretos que están promediados en la celda, la variable numérica fundamental en toda aplicación CFD. Esto distingue claramente al método de volúmenes finitos de otros métodos como diferencias finitas o elementos finitos, en los que la variable numérica fundamental es el valor local de la función en los nodos de la malla.

Tras la generación del mallado, el método de volúmenes finitos asocia un volumen finito

local, también denominado volumen de control, a cada punto de la malla, y a continuación aplica las leyes integrales de conservación a cada volumen local. Esta es otra gran diferencia con el método de diferencias finitas, donde el espacio discretizado se considera como un conjunto de puntos, mientras que en el método de volúmenes finitos el espacio discretizado está formado por un conjunto de pequeñas celdas, donde cada una de ellas está asociada a un nodo de la malla.

El método de volúmenes finitos tiene más ventajas con las mallas arbitrarias, en las cuales un gran número de opciones quedan abiertas para poder definir distintos volúmenes de control en los que imponer las leyes de conservación.

La posibilidad de modificar la forma y la localización de volúmenes finitos asociados a los nodos, así como las leyes y la precisión en la evaluación de los flujos a través de las superficies de control, proporciona una enorme flexibilidad al MVF. Esto explica la generalidad del método, aun cuando existan una serie de reglas que deben respetarse durante estas operaciones.

Una ventaja esencial del método de volúmenes finitos es que garantiza una discretización conservativa, que encaja perfectamente con el hecho de tener que resolver una ecuación de transporte conservativa. En las ecuaciones discretizadas es muy importante mantener la conservación global de las variables básicas del flujo: masa, momento y energía; por lo que será preciso respetar una serie de condiciones en la forma en que se lleva a cabo la discretización.

El método de volúmenes finitos tiene, por tanto, una ventaja decisiva: la discretización conservativa se satisface automáticamente con la discretización directa de la forma integral de las ecuaciones de gobierno.

3.1.1. Discretización numérica por el método de volúmenes finitos

En este apartado se van a resumir algunos conceptos básicos relacionados con el método de volúmenes finitos y sus implicaciones en la implementación práctica de un código de resolución.

3.1.1.1. Definiciones generales de la metodología numérica

La forma general de una ley de conservación para una cantidad escalar U, con fuentes volumétricas Q, sobre un volumen finito que incorpora flujos por las caras del volumen de control viene dada por (ver [6]):

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega + \oint_{S} \vec{J} \cdot d\vec{S} = \int_{\Omega} Q d\Omega \tag{3.1}$$

La ecuación general de transporte en su formulación integral ya había sido definida en el capítulo anterior como

$$\frac{\partial}{\partial t}\int_V \rho\phi dV + \oint_A (\rho\vec{v}\phi - \Gamma\nabla\phi)\cdot\vec{A} = \int_V S_\phi dV \tag{3.2}$$

Basta con identificar que $U = \rho\phi$, $\vec{J} = \rho\vec{u}\phi$ y $Q = S_\phi$, así como que el volumen se representa por Ω en lugar de V y el área por S en vez de A, para entender que estamos ante la misma expresión. En este capítulo se utilizará esta nueva nomenclatura introducida en la ecuación 3.1, pues es muy habitual su uso en la mayoría de los textos de referencia. Además, esta forma de expresar la ecuación de transporte es muy compacta e identifica de forma matemática el sentido de los términos involucrados: término temporal (unsteady), término de flujo por las superficies de control (fluxes) y término fuente.

La propiedad esencial de esta formulación es la presencia de la integral de superficie, así como el hecho de que la variación temporal de U dentro del volumen dependa únicamente de los flujos a través de esas superficies. Para observar que realmente se cumple que el esquema es conservativo, se procede a continuación a subdividir un volumen cualquiera en tres subvolúmenes (figura 3.1).

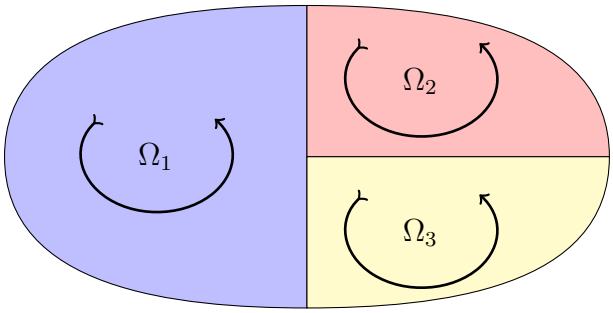


Figura 3.1: Leyes de conservación para los subvolúmenes Ω_1, Ω_2, Ω_3

Aplicando las leyes de conservación a cada subvolumen se tendría:

$$\begin{aligned}
\frac{\partial}{\partial t}\int_{\Omega_1} U\cdot d\Omega + \oint_{ABCA} \vec{J}\cdot d\vec{S} &= \int_{\Omega_1} Q\cdot d\Omega \\
\frac{\partial}{\partial t}\int_{\Omega_2} U\cdot d\Omega + \oint_{DEBD} \vec{J}\cdot d\vec{S} &= \int_{\Omega_2} Q\cdot d\Omega \\
\frac{\partial}{\partial t}\int_{\Omega_3} U\cdot d\Omega + \oint_{AEDA} \vec{J}\cdot d\vec{S} &= \int_{\Omega_3} Q\cdot d\Omega
\end{aligned} \tag{3.3}$$

Nótese que se puede recuperar la conservación global si simplemente se suman las tres ecuaciones en 3.3. Además, cuando se suman las integrales de superficie, las contribuciones de las líneas interiores ADB y DE siempre aparecen dos veces, pero con signos opuestos, evidenciando que los flujos internos se cancelan dos a dos. Por ejemplo, para el volumen Q, se tiene la contribución $\int_{DE} \vec{J}\cdot d\vec{S}$, mientras que para el volumen contiguo Ω_3 se tiene el término similar $\int_{ED} \vec{J}\cdot d\vec{S} = \int_{DE} \vec{J}\cdot d\vec{S}$, que se recorre en sentido opuesto. Por tanto, al hacer la suma, ambos términos se cancelan.

Esta propiedad esencial debe cumplirse en toda discretización numérica para afirmar que el esquema empleado es conservativo. Cuando esto no ocurre, la suma de las ecuaciones discretizadas sobre un número de volúmenes adyacentes contiene contribuciones de flujos interiores que aparecen como fuentes (numéricas) volumétricas internas. En ese caso, se dice que la discretización es no conservativa.

Nótese que si se cometen errores en la evaluación de los flujos en las fronteras interiores, la conservación global no se verá afectada.

3.1.1.2. Fundamentos del método de volúmenes finitos

El punto fuerte del método es su conexión directa con las propiedades físicas del flujo. De hecho, los fundamentos del método recaen en la discretización directa de la expresión integral de las leyes de conservación. Esto distingue claramente al método de volúmenes finitos de los métodos por diferencias finitas y elementos finitos, que discretizan la forma diferencial de las leyes de conservación. Para aplicar el método se debe subdividir el mallado obtenido de la discretización espacial en un número finito de volúmenes (celdas), quedando cada volumen de control asociado a cada uno de los puntos de la malla. Después se aplicará la ley de conservación en forma integral a cada uno de esos volúmenes finitos. Respecto a la forma en que se asocian los volúmenes de control (celdas) a los puntos del mallado, existen dos posibilidades:

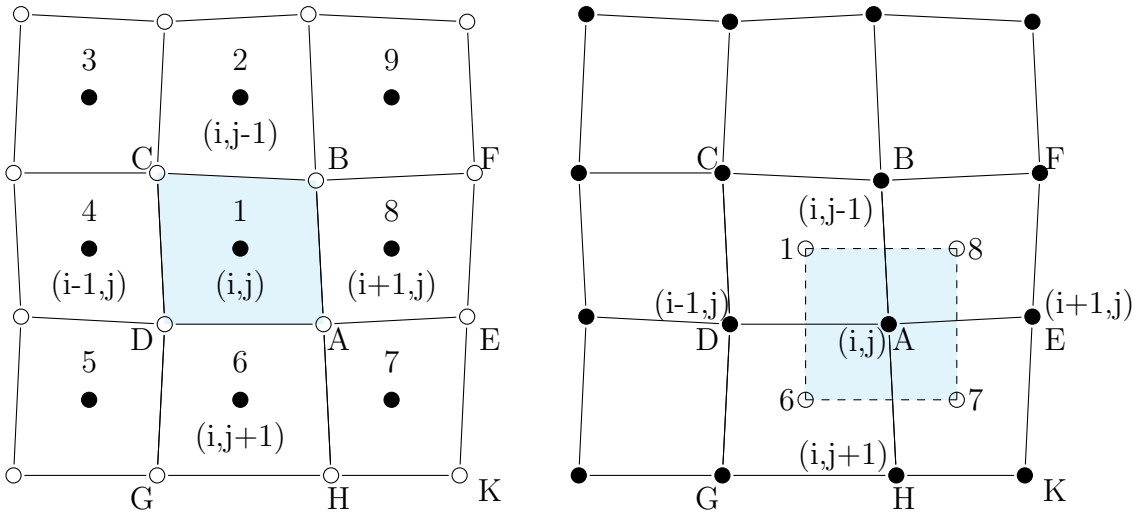
- Esquema basado en celdas (*cell-based* o *cell-centered approach*): en el que las variables se asocian (se almacenan) en los **centros** de las celdas y las líneas de las mallas definen los volúmenes finitos (celdas) y sus superficies. Es la elección obvia: hacer coincidir los volúmenes de control con las celdas. Las variables de flujo son valores promediados sobre cada celda y representativos, por ejemplo, del centroide de dicha celda.

- Esquema basado en nodos (nude-based, vertex-based o cell-vertex approach): en el que las variables se asocian en los vértices de la malla. Por tanto, las incógnitas están almacenadas en los puntos de la malla, es decir, en los vértices de las celdas. Este esquema es menos intuitivo que el anterior pero permite una mayor flexibilidad en la definición de los volúmenes de control.

Sea cual fuere la elección adoptada (normalmente el esquema basado en celdas, y por eso se habla indistintamente de celdas o de volúmenes de control), se debe garantizar que la suma de todos los volúmenes Ω_j (donde el subíndice j representa a cada una de las celdas del dominio) cubran el dominio discretizado y que no queden por tanto zonas vacías. A continuación se reemplaza en cada volumen de control Ω_j la ecuación integral 3.1 por su expresión discretizada, de forma que las integrales de volumen se expresan como los valores medios de las variables en el interior de las celdas y las integrales de superficie se reemplazan por la suma de los flujos en cada una de las caras de dicho volumen. Es decir:

$$\frac{\partial}{\partial t}(U_j \Omega_j) + \sum_{caras} \vec{J} \cdot \nabla \vec{S} = Q_j \Omega_j \tag{3.4}$$

La figura 3.2 muestra una discretización bidimensional típica centrada en las celdas (izquierda). En este caso (*cell-based)*, se identificaría la celda 1 *(i, j)* con el dominio Ω_j (superficie total ABCD en 2-D) y la variable U_j correspondería con $U_{i,j}$. El término de flujo debe sumar las contribuciones de los cuatro lados: AB, BC, CD y DA. La ecuación 3.4 es la expresión general del método de volúmenes finitos y debe ser el usuario el que defina, para cada volumen Ω_j seleccionado, cómo se definen las caras y volúmenes de cada elemento (esto es, cell-based o node-based) y cómo se calculan los flujos en las caras. Además, esta ecuación presenta una serie de características interesantes, que diferencia la interpretación del MVF de los métodos en diferencias o elementos finitos:



(a) Volumen de control centrado en la celda.

(b) Volumen de control centrado en los nodos.

Figura 3.2: Configuración del volumen de control centrado en celda y en nodo.

- Las coordenadas del punto J, asociadas a la localización exacta de U_j dentro del volumen de control, no aparecen explícitamente en la ecuación. Por lo tanto, U_j no está necesariamente anclado a un punto fijo dentro del volumen de control, sino que se puede considerar como un valor medio de la variable del flujo U en dicho volumen.

- Las coordenadas de los puntos de la malla aparecen únicamente en la obtención del volumen de la celda y de las áreas de las caras. Por ejemplo, en referencia a la figura 3.2, para la celda ABCD, que comprende el punto 1, sólo se requieren las coordenadas de A, B, C y D.

- Cuando no haya términos fuente, la formulación por MVF expresa que la variación del valor promediado de U sobre un intervalo de tiempo Δt es igual a la suma de

flujos que intercambia la celda considerada con sus vecinas. Para flujos estacionarios, la solución numérica se obtiene como resultado del balance de todos los flujos entrantes o salientes al volumen de control, es decir:

$$\sum_{caras} \vec{J} \cdot \Delta \vec{S} = 0 \tag{3.5}$$

- El método de volúmenes finitos permite una implementación directa de las condiciones de contorno, por ejemplo en los contornos sólidos en los que determinadas componentes normales han de ser cero. Así, en el caso de la ecuación de conservación de masa para la que $\vec{J} = \rho \vec{v}$, la condición de contorno en una pared será $\vec{J} \cdot d\vec{S} = 0$.

La ecuación 3.1, además de reemplazarse por su forma discreta, requiere de una integración temporal debido a la existencia del término no estacionario. Si se integra dicha ecuación desde el instante anterior $(n-1)\Delta t$ hasta el instante final $n\Delta t$ para cada volumen de control Ω_j asociado a la celda J se obtiene:

$$\int_{\Omega_j} U \cdot d\Omega j|^n = \int_{\Omega_j} U \cdot d\Omega j|^{n-1} - \sum_{caras} \int_{n-1}^{n} \left(\vec{J} \cdot \Delta \vec{S} \right)_c dt + \int_{n-1}^{n} dt \int_{\Omega_j} Q \cdot d\Omega_j \tag{3.6}$$

Introduciendo la variable promedio en el interior de la celda, $\overline{U}_j^{n-1}$, así como la fuente $\overline{Q}_j^{n-1}$, ambas evaluadas en el instante $(n-1)\Delta t$; y definiendo las cantidades promediadas en el tiempo para el volumen de control como Q_j^* para la fuente y $\vec{J}^*$ para el flujo numérico se tiene:

$$\overline{U}_j^{n-1} = \frac{1}{\Omega_j} \int_{\Omega_j} U d\Omega|^{n-1} \vec{J}^* \cdot \Delta \vec{S} = \frac{1}{\Delta t} \int_{n-1}^{n} \vec{J} \Delta \vec{S} dt \tag{3.7}$$

$$\overline{Q}_j^{n-1} = \frac{1}{\Omega_j} \int_{\Omega_j} Q d\Omega \overline{Q}_j^* = \frac{1}{\Delta t} \int_{n-1}^{n} \overline{Q}_j dt \tag{3.8}$$

Por tanto, la discretización conservativa toma la forma:

$$\overline{U}_j \Omega_j|^n = \overline{U}_j \Omega_j|^{n-1} - \Delta t \sum_{caras} \vec{J}^* \cdot \Delta \vec{S} + \Delta t \overline{Q}_j^* \Omega_j \tag{3.9}$$

La ecuación 3.9 es una **relación exacta** para la evolución temporal de las variables conservativas U_j^{n-1} y promediadas en el interior de la celda I. Obsérvese nuevamente cómo no hay ningún punto de la malla asociado a U_j^{n-1} (la variable está únicamente anclada a la celda I). La forma en que se calcule el flujo numérico $\vec{J}^*$ será la que permita identificar el esquema de discretización empleado (v. ejemplos en 4.4), pues la definición de $\vec{J}^*$ es la que permite aproximar el flujo real promediado en el tiempo para cada cara de una celda.

Es evidente que en aras de satisfacer el principio de conservación a este nivel discreto, la estimación del flujo numérico en una determinada cara de una celda debe ser independiente de la celda a la que pertenece.

La ausencia en la ecuación 3.9 de un índice temporal tanto en el sumatorio de los flujos numéricos como en el término fuente indica que es posible elegir en qué instante se quieren evaluar: bien en el instante anterior *(n - 1)* **-esquema explícito-**, bien en instante actual *n* **-esquema implícito-** como se verá en el próximo apartado.

3.2. Herramienta numérica OpenFOAM

OpenFOAM es en primer lugar, y ante todo, una *librería de C++* usada principalmente para crear ejecutables conocidos como *aplicaciones*. Estas aplicación se separan en dos categorías: los *solvers*, que están diseñados cada uno para resolver un problema específico dentro de la mecánica de los medios continuos; y las utilidades, que están diseñadas para mejorar las tares que incluyen el manejo de datos. La distribución de OpenFOAM contiene gran cantidad de *solvers* y *utilidades* que cubren un gran abanico de problemas. Uno de los puntos fuertes que tiene OpenFOAM es que los usuarios pueden aportar nuevos *solvers* y *utilidades*, siempre y cuando tenga los conocimientos de programación y de física necesarios para ello.

3.2.1. Estructura de archivos del estudio

El estudio de un caso particular usando la herramienta OpenFOAM requiere de una carpeta principal y un mínimo de tres subcarpetas. La carpeta principal puede tener cualquier nombre decidido por el usuario, normalmente, coincide con el nombre de un caso similar encontrado en los tutoriales del OpenFOAM. Por otro lado las subcarpetas que se hallan dentro de la carpeta de nuestro estudio tienen nombres que no pueden ser alterados, de lo contrario, OpenFOAM no podrá encontrarlas al ser ejecutado, y por tanto, éste se detendrá mostrando por pantalla un mensaje de error.

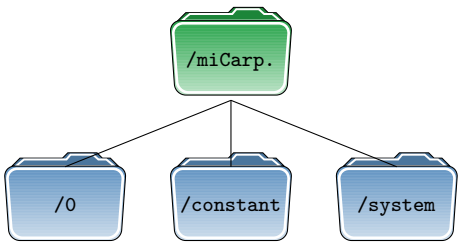


Figura 3.3: Organización de la carpeta con el estudio.

Esta carpeta a su vez alberga a otras tres subcarpetas que serán las que guarden la configuración del estudio, tanto condiciones iniciales como esquemas numéricos, algoritmos de acoplamientos etc. Estas subcarpetas son:

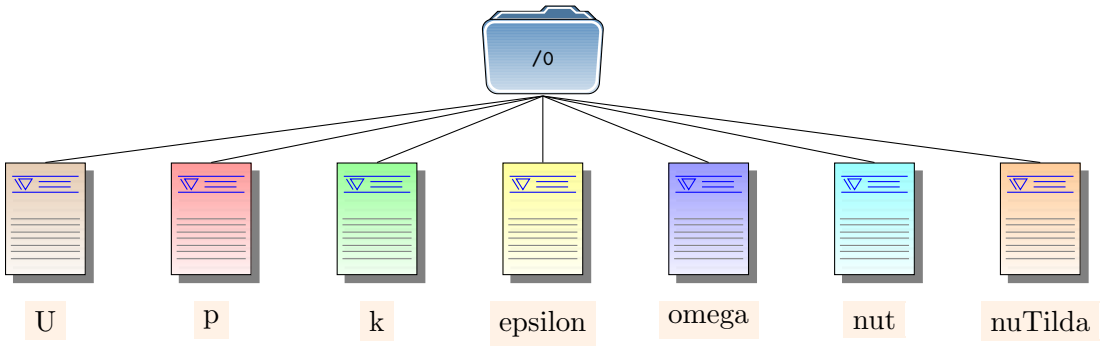


Figura 3.4: Organización de la carpeta las condiciones iniciales.

- Carpeta *0*: En ella se configura las condiciones iniciales del problema en archivos separados, cada variable viene definida por un archivo con su mismo nombre.
- Carpeta *constant*: En esta carpeta se configura en modelo de turbulencia como el valor de la viscosidad. En su interior se encuentra la carpeta *poliMesh* quien almacena la información relevante a la construcción y geometría de la malla por medio del archivo *blockMeshDict*. El archivo *blockMeshDict.m4* es un archivo opcional que nos ayuda, en caso de así requerirlo, a construir la malla de manera paramétrica, ya que *blockMeshDict* por sí mismo no contempla esta opción. Esto se hace mediante la instrucción *m4 blockMeshDict.m4 > blockMeshDict*.

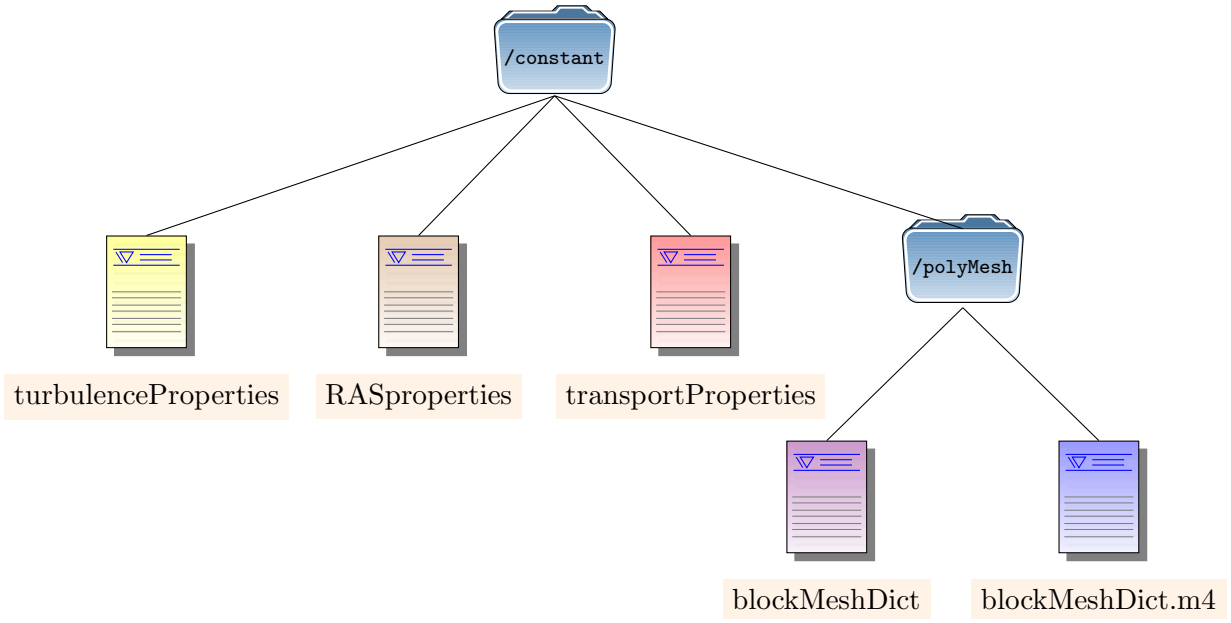


Figura 3.5: Organización de la carpeta *constant*.

- Carpeta *system*: Esta es la carpeta donde definiremos los esquemas, solver, tiempo de ejecución, paso de tiempo, interpolaciones, etc, de nuestro estudio.

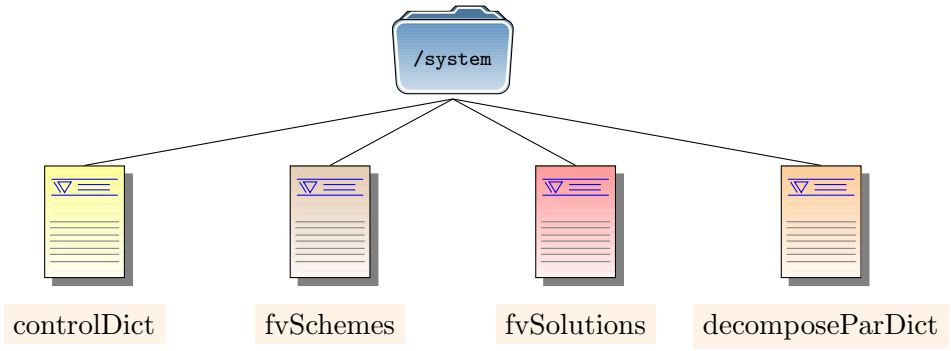


Figura 3.6: Organización de la carpeta *system*.

3.2.2. Ejecución de una simulación

Cada aplicación está diseñada para ser ejecutada desde la terminal de Linux, típicamente leyendo y escribiendo un conjunto de datos asociados a un caso en particular.

Para cualquier aplicación podemos encontrar su descripción escribiéndola en la consola de comandos seguido del argumento **-help**. Por ejemplo,

```
blockMesh -help
```

que devuelve el mensaje,

```
Usage: blockMesh [-region region name] [-case dir] [-blockTopology]
[-help] [-doc] [-srcDoc]
```

Los argumentos entre corchetes, [ ], son opcionales. Si la aplicación es ejecutada desde el interior de un caso, OpenFOAM operará dentro de él. Alternativamente, la opción **-case** <**caseDir**> permite a la aplicación ejecutarse desde cualquier directorio.

Como cualquier otro ejecutable UNIX/Linux, las aplicaciones se pueden ejecutar en segundo plano,esto es, no es necesario esperar a que la aplicación haya terminado de ejecutarse para poder seguir utilizando el terminal con otros comandos. Si quisiéramos ejecutar **blockMesh** en segundo plano y almacenar la salida en un fichero log, escribiríamos:

```
blockMesh > log &
```

3.2.3. Ejecución de una simulación en paralelo

Esta sección describe cómo ejecutar OpenFOAM en paralelo. El método que utiliza OpenFOAM para el cálculo en paralelo es conocido como *descomposición del dominio*, en el cual la geometría se despieza en varias partes, enviando cada una a un procesador distinto. El proceso del cálculo en paralelo implica: descomposición del dominio (malla y campos); ejecución en paralelo del caso; reconstrucción del dominio y post-procesado. Para la ejecución en paralelo se necesita la implementación estándar de dominio público **openMPI**.

3.2.4. Descomposición de la malla y campos iniciales

Tanto la malla como los campos escalares y vectoriales se descomponen haciendo uso de la utilidad **decomopsePar**. Su principal propósito es descomponer el dominio con el mínimo esfuerzo pero de tal manera que garantice una buena solución. Geometría y campos se descomponen de acuerdo con un conjunto de parámetros que se encuentran en un diccionario llamado **decomposeParDict** que puede ser copiado desde el directorio **interFoam/icoFoam** del tutorial.
Un ejemplo es:

```
numberOfSubdomains 8;

method simple;

simpleCoeffs
{
  n (2 4 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();
```

Aquí se puede elegir entre cuatro métodos de descomposición, especificada por la palabra clave **method** como se describe a continuación.

- *simple*. Descomposición simple en el cual el dominio es dividido en distintos subdominios siguiendo una dirección, por ejemplo dos subdominios siguiendo la dirección x, cuatro subdominios siguiendo la dirección y, y un único subdominio en la dirección z.

- *hierarchical*. Descomposición geométrica jerarquizada, la cual es igual que el método simple, excepto que en esta ocasión el usuario puede elegir en qué orden se realiza la división del dominio, por ejemplo, primero en el eje x, seguido del eje y, y finalizando en el eje z.

- *scotch*. En esta descomposición no se requiere de ninguna entrada sobre la geometría por parte del usuario e intenta minimizar el número de contornos a procesar. El

usuario puede especificar unos pesos para la descomposición entre procesadores, **processorWeights**, si los procesadores poseen rendimientos distintos entre ellos. También está la palabra reservada **strategy** que controla la descomposición a través de una string que se suministra a la entrada **scotch**.

- *manual*. Descomposición manual, donde el usuario directamente especifica qué celda se asigna a qué procesador.

La utilidad **decomposePar** se llama desde consola tecleando simplemente en la carpeta donde se ejecuta el estudio.

```
decomposePar
```

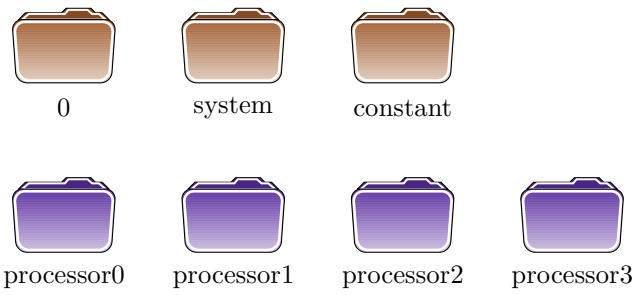


Figura 3.7: Vista de la carpeta una vez descompuesto el dominio en cuatro subdominios. Las carpetas violeta contienen los datos pertenecientes a cada subdominio, tanto la malla como los valores iniciales de los campos escalares y vectoriales; como sus condiciones de contorno.

3.2.5. Ejecución de un caso descompuesto

Para ejecutar un caso ya preparado en paralelo usando **mpirun** escribimos en consola

```
mpirun --hostfile <machines> -np <nProcs>
<foamExec> <otherArgs> -parallel > log &
```

donde <nProcs> es el número de procesadores; <foamExec> es el ejecutable, por ejemplo **icoFoam**; y, la salida es redirigida a un fichero llamado **log**.

3.2.6. Post-procesado de un caso descompuesto

Después de que una simulación haya terminado de ejecutarse en paralelo se debe recomponer los datos almacenados en las carpetas **processorX** donde X es un número que va desde 0 hasta el número de subdominios en los que hayamos dividido la malla menos 1,

$$0 \leq X \leq numProc. - 1$$

Esto se realiza con el comando

`reconstructPar`

Una vez reconstruido el caso, podemos visualizar el resultado de nuestro estudio en **paraView** llamando desde la consola a la función

`paraFoam`

Éste creará un archivo temporalmente con la extensión *.OpenFOAM* con los datos de la simulación en su interior que puede ser ya interpretado por el programa **paraView**.

Figura 3.8: Vista de las nuevas carpetas creadas una vez reconstruida la descomposición donde se ha realizado una simulación de un segundo con un paso temporal de 0.1 segundos. Las carpetas amarillas contienen los datos de los campos escalares y vectoriales para cada paso de tiempo, mientras que la carpeta *posProcessing* almacena en un fichero de texto plano las fuerzas medidas sobre el cilindro. Esta última carpeta se crea durante el proceso de simulación.

3.3. Funciones de muro (wall function)

3.3.1. Funciones de muro estándar

El primer punto para el estudio y desarrollo de la Wall Function Boundary Condition (WFBC) es recordar la estructura teórica de la capa límite, resumida en la figura 3.9, donde se definen;

- Coordenada normal adimensional. Para diferenciar cada una de las subcapas se utiliza el parámetro adimensional y^+ que se define como:

$$y^+ = \frac{u_\tau y}{\nu} \tag{3.10}$$

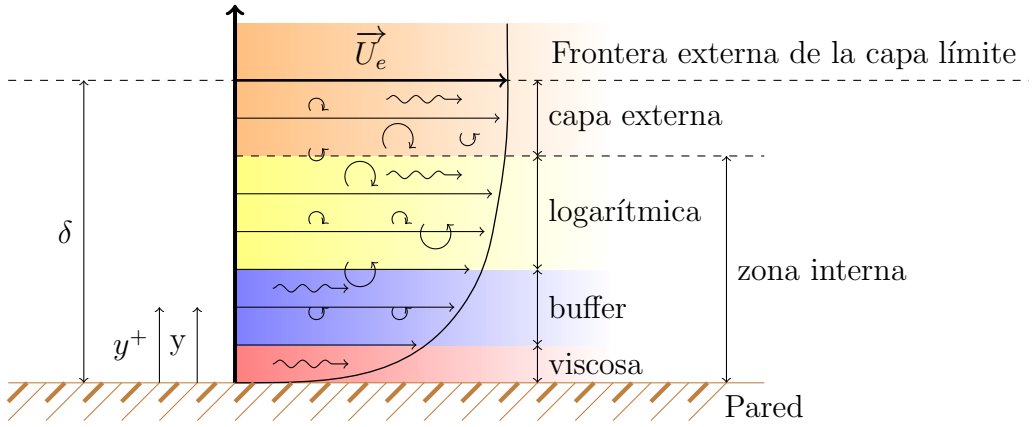


Figura 3.9: Estructura de la capa límite

donde u_τ es la velocidad de fricción.

- Velocidad de fricción en la pared.

$$u_\tau = \sqrt{\frac{\tau_{pared}}{\rho}} \Longrightarrow (\tau_{pared} = u_\tau^2 \rho) \tag{3.11}$$

siendo τ la tensión cortante en la pared.

- Velocidad tangencial adimensional.

$$u^+ = \frac{\overline{u}}{u_\tau} = f(y^+) \tag{3.12}$$

(Ley de la pared)

- Subcapa viscosa $(y^+ < 5, Re = \dfrac{\overline{u}}{u_\tau} \lesssim 1) \Longrightarrow$ fuerzas viscosas predominan. En este caso podemos decir que

$$\tau(y) \approx cte / \tau(y) = \mu \frac{\partial \overline{u}}{\partial y} \approx \tau_{pared} \longrightarrow \forall\ 0 \leq y^+ \leq 5 \tag{3.13}$$

Integrando, y teniendo en cuenta que $\overline{u}(y^+ = 0) = 0$

$$\overline{u} = \frac{\tau_{pared}\, y}{\mu} \Rightarrow u^+ = \frac{\overline{u}}{u_\tau} = \frac{\tau_{pared}\, y}{\mu u_\tau} = \frac{u_\tau^2 y}{\frac{\mu}{\rho} u_\tau} = y^+ \Longrightarrow u^+ = y^+ \tag{3.14}$$

(Variación lineal)

- Subcapa logarítmica ($30 < y^+ < 300$) Aquí el flujo es dominado por las fuerzas de inercia, y las tensiones turbulentas son más importantes que las viscosas. Aquí se demuestra que:

$$\frac{\partial u^+}{\partial y^+} = \frac{1}{\kappa y^+} \Longrightarrow u^+ = \frac{1}{\kappa} ln(E y^+) \tag{3.15}$$

donde $\kappa = 0{,}41$ y E=9.793 para paredes suaves.

Para ver cómo varía $\tau(y)$

$$\begin{aligned} \frac{\partial u^+}{\partial y^+} = \frac{1}{\kappa y^+} \Longrightarrow \frac{1}{u_\tau}\frac{\partial \overline{u}}{\partial y^+} = \frac{1}{u_\tau}\frac{1}{\frac{u_\tau}{u}}\frac{\partial \overline{u}}{\partial y} = \frac{1}{\kappa y^+} \\ \frac{\mu}{\tau_{pared}}\frac{\partial \overline{u}}{\partial y} = \frac{1}{\kappa y^+} \Longrightarrow \frac{\tau(y)}{\tau_{pared}} = \frac{1}{\kappa y^+} \\ \tau(y) = \frac{\tau_{pared}}{\kappa y^+} \end{aligned} \tag{3.16}$$

En resumen; podemos esquematizar la capa límite como se aprecia en la figura 3.10;

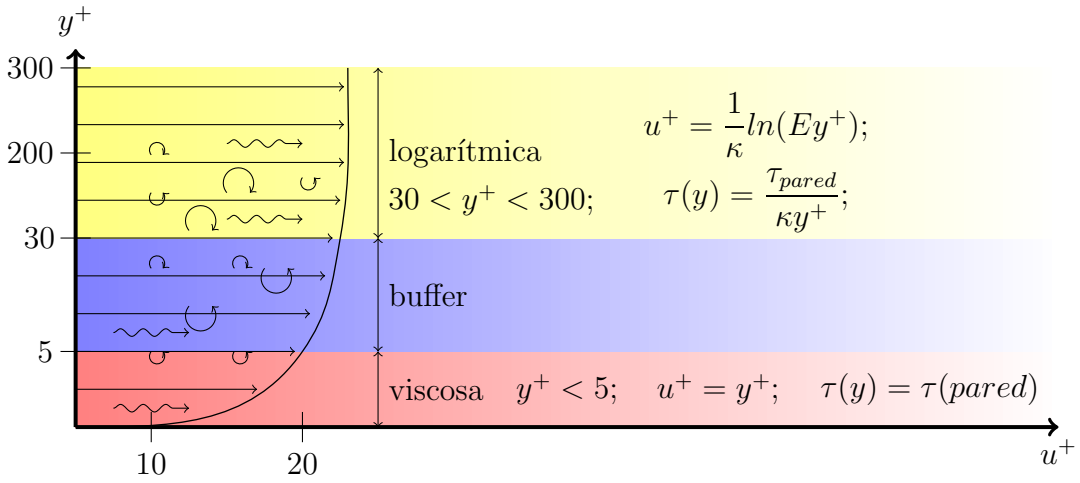


Figura 3.10: Estructura de la capa límite.

Como se ve, la variación de $\overline{u}$ es muy grande cerca de la pared, por lo que resolver este problema numéricamente directamente implicaría discretizar adecuadamente esta zona, haciendo que las celdas cercanas al contorno sean de tamaño no mayor a $y^+ = 1$.

Otra posibilidad es aprovechar el conocimiento del comportamiento de las variables a lo largo de la capa límite para modelarla en lugar de discretizarla, y así, reducir sensiblemente el esfuerzo computacional (número de celdas y paso de tiempo).

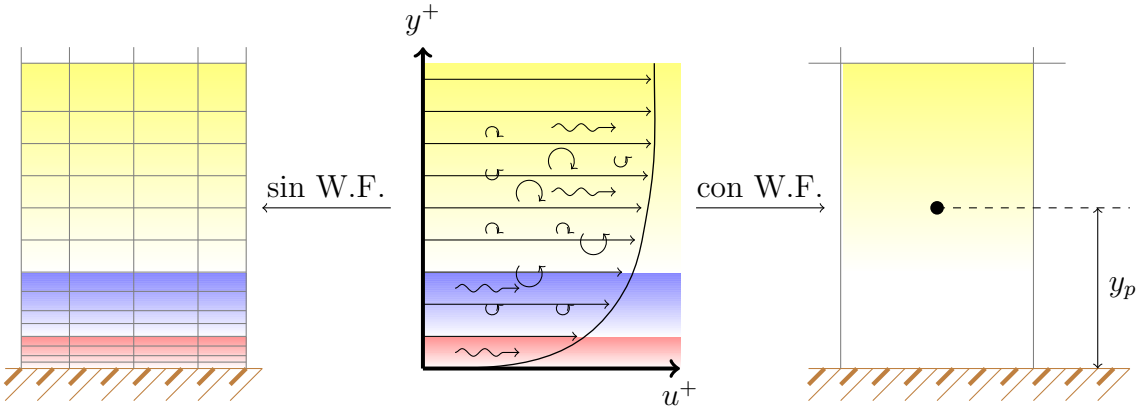


Figura 3.11: Discretización *con* y *sin* Wall Function

Una forma de modelar la capa límite es aplicar condiciones de contorno, no sobre el contorno físico, sino a una distancia del mismo, donde los valores de la variable a esa distancia dependan de las condiciones de contorno físico, del flujo y del modelo de la capa límite.

Para poder definir concretamente cómo llevar esto a cabo, lo primero es recordar que, si se asume la hipótesis de Boussinesq, por la cual las tensiones de Reynolds $\overline{\mu_i'\mu_j'}$ pueden ser relacionadas directamente con las tasas de deformación en un fluido Newtoniano a través de una expresión equivalente a la utilizada para flujos Newtonianos laminares, pero utilizando una nueva viscosidad turbulenta variable $\nu_t = \nu_t(x_i, t)$ de manera que:

$$-\overline{\mu_i'\mu_j'} = \nu_t \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{3.17}$$

Entonces, las ecuaciones de conservación del momento de Navier-Stokes son idénticas a las obtenidas para un flujo laminar pero utilizando $\overline{u_i}$ (velocidad media) y $\nu_{eff} = \nu + \nu_t(x_i, t)$ en lugar de ν_i y U, y con el término $\overline{p} + \frac{2}{3}\rho k$ en lugar de p

En este flujo turbulento, además a las ecuaciones de Navier-Stokes se le acoplan unas ecuaciones de transporte que permitan el cálculo de ν_t en cada punto, y que en el caso del modelo de turbulencia $k - \epsilon$ son dos ecuaciones, una para la energía cinética turbulenta k y otra para la tasa de disipación ϵ. Una vez resueltas estas ecuaciones para un instante de tiempo dado, la viscosidad turbulenta a cada punto se obtiene a partir de dichos campos escalares k y ϵ como:

$$\nu_t = \frac{C_\mu k^2}{\epsilon} \tag{3.18}$$

ecuación que encierra el modelo $k - \epsilon$ y donde $C_\mu = 0{,}09$

Así, en cada instante de tiempo, la solución puede obtenerse de manera iterativa en dos fases: en primer lugar, se resuelven los campos de presiones y velocidades utilizando los valores de ν_t y k obtenidos en el aso anterior. Una vez que $\overline{u}$ y $\overline{p}$ han convergido, se resuelven las ecuaciones de transporte para ϵ y k utilizando los nuevos campos $\overline{u}$ y $\overline{p}$, pudiéndose entonces calcular el nuevo valor del campo $\nu_t = \nu_t(x_i, t)$. En caso de que el campo ν_t haya variado sensiblemente, todo el proceso puede repetirse con los nuevos campos ν_t y k antes de pasar al siguiente paso de tiempo.

Así, para cada conjunto de ecuaciones y para cada variable, distintos tipos de contorno pueden ser especificados. La estrategia más utilizada consiste en establecer la tasa de disipación ϵ_p en un punto P de la subcapa logarítmica (nodo P) además de calcular la tensión tangencial. En principio, podría también establecerse la velocidad u_p a dicho punto, pero ésta resulta ser una estrategia menos robusta numéricamente y por eso no se utiliza.

A continuación se derivan las expresiones correspondientes a la denominada función de pared estándar (standard wall function), derivada por Prandtl en 1904. Es importante destacar que exista muchas otras funciones de pared, y que además dependerán de los parámetros utilizados por el modelo de turbulencia (ver por ejemplo Mellr 1966, Nakagoma 1984 o Cruz y Silva-Freiz 1998)

3.3.2. Formulación de la WFBC standard

Nos queda ahora obtener las expresiones de la WFBC para ser aplicadas en $y = y_p$ en la subcapa logarítmica, (para la que se derivan las expresiones) por lo que será necesario comprobar a -*posteriori*- que el primer centro de la celda se encuentra en dicha capa ($30 < y^+ < 300$).

La obtención de dichas expresiones parte de la siguiente hipótesis: La producción y la disipación de energía turbulenta están en equilibrio en la subcapa logarítmica, hipótesis derivada de las observaciones y simulaciones DNS en la capa límite.

De la ecuación de gobierno para la energía cinética turbulenta, se deriva que el término de producción (fuente) es:

$$P = -\overline{u_1' u_2'} \frac{\partial \overline{u}}{\partial y} \tag{3.19}$$

por lo que, dado que $\epsilon = P$ para ($30 < y^+ < 300$), podemos escribir:

$$\epsilon = -\overline{u_1' u_2'} \frac{\partial \overline{u}}{\partial y} \tag{3.20}$$

donde:

$$\frac{\partial \overline{u}}{\partial y} = \frac{\partial \overline{u}}{\partial u^+} \cdot \frac{\partial u^+}{\partial y} = u_\tau \left(\frac{\partial u^+}{\partial y^+} \cdot \frac{\partial y^+}{\partial y} \right) = u_\tau \left(\frac{1}{k y^+} \frac{u_2}{u} \right)$$

$$= u_\tau \left(\frac{1}{k\dfrac{u_\tau y}{u}} \frac{u_\tau}{u} \right) = \frac{u_\tau}{ky} \tag{3.21}$$

donde de nuevo se ha utilizado expresiones válidas tan sólo para la capa logarítmica.
Por otro lado, según la ecuación (3.17), en el entorno de la pared se puede escribir:

$$-\overline{u_1' u_2'} = \nu_t \left(\frac{\partial \overline{u_1}}{\partial y} + \overbrace{\frac{\partial \overline{u_2}}{\partial x}}^{=0} \right) = \nu_t \frac{\partial \overline{u}}{\partial y} = \frac{\tau_{pared}}{\rho} \tag{3.22}$$

que, según la ecuación (3.11), puede expresarse también como:

$$-\overline{u_1' u_2'} = u_\tau^2 \tag{3.23}$$

Así, sustituyendo (3.21) y (3.23) en (3.20), se puede expresar la tasa de disipación en la subcapa logarítmica como:

$$\epsilon = \frac{u_\tau^3}{ky} \tag{3.24}$$

Por otro lado, sustituyendo (3.18) y (3.21) en (3.22), y recordando (3.11), puede escribirse también:

$$u_\tau^2 = \frac{\tau_{pared}}{\rho} = u_\tau \frac{\partial \overline{u}}{\partial y} = \frac{C_\mu \kappa^2}{\epsilon} \frac{u_\tau}{\kappa y} \tag{3.25}$$

del primer y último término de la ecuación podemos también despejar otra expresión para la disipación:

$$\epsilon = \frac{C_\mu k^2}{u_\tau \kappa y} \tag{3.26}$$

Despejando de (3.24) y sustituyendo en (3.26):

$$\epsilon = \frac{C_\mu \kappa^2}{ky} \frac{1}{(\epsilon \kappa y)^{1/3}}$$
$$\epsilon^{4/3} = \frac{C_\mu k^2}{(\kappa y)^{4/3}}$$

$$\epsilon = \frac{C_\mu^{3/4} k^{3/2}}{\kappa y} \tag{3.27}$$

Esta última es la expresión que puede utilizarse como condición de contorno del tipo Dirichlet para especificar la tasa de disipación en $y = y_p$, si se conoce de la iteración anterior la energía cinética turbulenta en dicho punto k_p. Así, la condición de contorno será:

$$\epsilon = \frac{C_\mu^{3/4} k_p^{3/2}}{\kappa y} \tag{3.28}$$

mientras que la condición de contorno para la energía cinética turbulenta será de gradiente normal nulo en el contorno físico:

$$\left.\frac{\partial k}{\partial n}\right|_{n=0}$$

Aquí es interesante también ver que, de la comparación de (3.24) y (3.27) se extrae que la velocidad de fricción puede expresarse en la subcapa logarítmica como función de la energía cinética turbulenta:

$$u_\tau = C_\mu^{1/4} k^{1/2} = u_\tau(k) \tag{3.29}$$

Por otro lado, la producción en este punto puede escribirse, teniendo en cuenta (3.19), (3.22), (3.21) y (3.29) como:

$$\begin{aligned} P &= -\overline{u_1' u_2'} \frac{\partial \overline{u}}{\partial y} = \nu_t \frac{\partial \overline{u}}{\partial y} \frac{\partial \overline{u}}{\partial y} = \nu_t \frac{u_\tau}{\kappa y} \frac{\partial \overline{u}}{\partial y} \\ P_p &= \nu_t \frac{C_\mu^{1/4} k^{1/2}}{\kappa y} \left.\frac{\partial u}{\partial y}\right|_{y=y_p} \end{aligned} \tag{3.30}$$

y sustituyendo (3.29) en (3.10), se puede expresar y^+ como

$$y^+ = \frac{C_\mu^{1/4} k_P^{1/2} y_P}{\nu} \tag{3.31}$$

y también utilizando (3.29), se puede expresar la tensión tangencial en la pared como:

$$\tau_{pared} = \rho u_\tau^2 = \rho u_\tau u_\tau \frac{\overline{u_P}}{\overline{u_P}} = \rho u_\tau \frac{\overline{u_P}}{u_P^+} = \rho C_\mu^{1/4} k^{1/2} \frac{\overline{u}}{u^+}$$

que, dado que es en la subcapa logarítmica, puede escribirse como:

$$\tau_{pared} = \rho \frac{C_\mu k_P^{1/2} u_P \kappa}{ln(Ey^+)} \tag{3.32}$$

La última variable a calcular es la viscosidad turbulenta. Para ello, partimos de la expresión:

$$\nu_{eff} \frac{\partial \overline{u}}{\partial n} = (\nu + \nu_t) \frac{\partial \overline{u}}{\partial n} = \frac{\tau_w}{\rho} \tag{3.33}$$

de donde:

$$\nu_t = \frac{\tau_{pared}}{\rho} \left(\frac{\partial \overline{u}}{\partial n} \right)^{-1} - \nu \tag{3.34}$$

Despejando $C_\mu^{1/4} k_P^{1/2}$ de (3.31), sustituyéndolo en (3.32), sustituyendo entonces el resultado en (3.34) y expresando el gradiente de manera discreta como $\frac{\partial \overline{u}}{\partial n} = \frac{\overline{u_P}}{y_P}$, se llega a:

$$\nu_t = \nu \left(\frac{y^+ \kappa}{ln(Ey^+)} - 1 \right) \tag{3.35}$$

que es la expresión del tipo $\nu_t = \nu_t(y^+)$ en la subcapa logarítmica.

3.3.3. Implementación de la Wall Function en OpenFOAM

Las *Wall Function* están implementadas como condiciones de contorno junto a la resolución de ecuaciones adicionales para el cierre de ecuaciones mediante el cálculo de la viscosidad turbulenta. Estas ecuaciones adicionales están implementadas en ficheros con el nombre del modelo, por ejemplo, *kEpsilon.C*, la cual resuelve primero la ecuación de disipación, añadiendo restricciones para fijar ϵ_p, y resuelve después la ecuación de la energía turbulenta para finalmente calcular el nuevo campo ν_t (función *correct*).

La ϵ_p está implementado en *epsilonWallFunctionFvPatchScalarField.C,* en la función *calculate* y la condición de contorno para k está implementada en *kqRWallFunctionFvPatchScalarField.C*, y es tan sólo una condición de contorno donde se le aplica un gradiente de k igual a cero *zeroGradient*. ϵ_p se fija en los puntos p mediante la función *boundaryManipulate()*, que implementa en *fvMatrix.C* que llama a *manipulateMatrix()*, implementada a su vez en el modelo, por ejemplo, en *kEpsilon.C*. Esta función fija los valores de la solución en ciertos puntos mediante la función *setValues.C*.

La condición de contorno para ν_t, a partir de (2.30), se establece en *nutKWallFunctionFvPatchScalarField.C*, en la función '*calcnut*', y siempre que se esté por encima de la subcapa viscosa. En la subcapa viscosa, se asume $\nu_t = 0$.

La llamada a la resolución de las ecuaciones de transporte del modelo turbulento (ver tercer punto) se realizarán a través de *turbulence* $\longrightarrow$ *correct*(), siendo *correct()* una función virtual que se define en tiempo de ejecución en función del modelo elegido.

Además puede hacerse que las funciones del modelo de turbulencia entren también en las iteraciones y su convergencia sea tenida en cuenta. Para ello, debe utilizarse *pimpleFoam* y establecer *turbOnFinalIterOnly* como '*off*'.

3.3.4. Cálculo de y^+ en OpenFOAM

Para el cálculo de y^+ se ha valido del código aportado por el usuario de CFD-Online [11] Daniel Wei. De este código sólo nos centraremos en aquellas líneas donde se resuelve tanto el valor de y^+ como el valor de u_τ. Veamos la parte del código que nos interesa:

```
(...)
forAll(patches, patchi)
        {
            const fvPatch& currPatch = patches[patchi];

            if (isA<wallFvPatch>(currPatch))
            {
                if (compressible)
                {
                    (...)
                }
                else
                {
                    // wallDistanceOnPatch.boundaryField()[patchi] = d[patchi];
                    yPlus.boundaryField()[patchi] =
                        d[patchi]
                       *sqrt
                        (
                            nuEff.boundaryField()[patchi]
                           *mag(U.boundaryField()[patchi].snGrad())
                        )
                       /nuLam.boundaryField()[patchi];

                    uTau.boundaryField()[patchi] =
                        sqrt
                        (
                            nuEff.boundaryField()[patchi]
                           *mag(U.boundaryField()[patchi].snGrad())
                        );
                }

                const scalarField& Yp = yPlus.boundaryField()[patchi];

                // Mean uTau estimation
                uTauAvg.value() += average(uTau.boundaryField()[patchi]);
                nPatch ++;

(...)

            }
        }
(...)
```

A través de un bucle *for* se recorren todas las celdas, *patches*, que colindan con la superficie de tipo *Wall* donde se desea calcular y^+ y u_τ, esto es en nuestro caso las celdas adyacentes al cilindro. Dentro del bucle aparece una bifurcación *if-else* donde se separa los cálculos de y^+ y u_τ en función de si el fluido es compresible o no. Puesto que estamos estudiando el caso de un fluido incompresible se ha eliminado la parte del *if-else* donde se evalúa este caso, dejando intacto el caso incompresible que es el que estamos analizando.

La definición de y^+ aparece en la ecuación 3.10 que repetimos a continuación

$$y^+ = \frac{u_\tau y}{\nu}$$

donde u_τ viene definida en 3.11 que repetimos a continuación para mayor comodidad del lector.

$$u_\tau = \sqrt{\frac{\tau_{pared}}{\rho}}$$

Puesto que en la subcapa viscosa se cumple que

$$\tau(y) = \mu \frac{\partial \overline{u}}{\partial y}$$

Combinando las ecuaciones tenemos que

$$u_\tau = \sqrt{\frac{\tau_{pared}}{\rho}} = \sqrt{\frac{\mu \frac{\partial \overline{u}}{\partial y}}{\rho}} = \sqrt{\nu \frac{\partial \overline{u}}{\partial y}} \tag{3.36}$$

La ecuación 3.36 la podemos ver implementada en el código donde:

- u_{tau} es *uTau.boundaryField()[patchi]*
- ν es *nuEff.boundaryField()[patchi]*
- $\frac{\partial \overline{u}}{\partial y}$ es *mag(U.boundaryField()[patchi].snGrad())*

Con este valor de u_{tau} es fácil calcular y^+ desde la ecuación 3.11 sabiendo que:

- y es la distancia normal desde el nodo perteneciente a la celta adyacente a una superficie a dicha superficie.
- ν es la viscosidad del fluido en la subcapa viscosa, esto es, *nuLam.boundaryField()[patchi];*

3.3.5. Tratamiento automático de paredes

Las funciones de pared estándar no son siempre deseables ya que con ellas se elimina la influencia de la subcapa viscosa. Especialmente para flujos a bajo número Reynolds, o cuando estamos interesados en realizar un estudio sobre la transferencia de calor de paredes a fluidos, la omisión de esta subcapa puede tener un impacto muy significativo en la solución. Por ejemplo, el ratio de flujo másico en una tubería, o para flujos en canales a bajos números de Reynolds puede acarrear un error de un 10 % y más, debido a la relativamente alta influencia de la porción viscosa de la capa límite. Para flujos externos, el efecto es generalmente menor, ya que el desplazamiento de flujos externos tiene un efecto de segundo orden.

Sin embargo, es deseable tener una formulación robusta y precisa en la formulación de las subcapas para ser capaces de resolver las ecuaciones a lo largo de toda la superficie. Los modelos basados en low-Re $k-\epsilon$ generalmente no resuelven los requisitos impuestos por la complejidad de los fluidos industriales. El siguiente método está, por lo tanto, basado en la formulación del modelo $k-\omega$ para paredes. Mientras que el tratamiento de las ecuaciones de ω son conocidas, la formulación óptima podría evitar los requisitos de una malla rígida. La idea detrás del tratamiento automático de paredes es que el modelo se mueva gradualmente entre la subcapa viscosa y la subcapa logarítmica basándonos en la densidad de malla. La ecuación de ω es muy buena para esta tarea, ya que ella nos provee de una solución analítica tanto para la subcapa viscosa como para la logarítmica. Para esto se define una función de mezcla (*blending function*) dependiendo del valor de la y^+.

Las soluciones para ω en las subcapas viscosa y logarítmica en paredes son (ver [3][4]):

$$\omega_{visc} = \frac{6\nu}{0{,}075y^2};\ \omega_{log} = \frac{1}{0{,}3\kappa}\frac{u_\tau}{y} \tag{3.37}$$

Éstas pueden ser reformuladas en términos de la y^+ mediante una función de mezcla.

$$\omega_1(y^+) = \left(\omega^2_{visc}(y^+) + \omega^2_{log}(y^+)\right)^{0,5} \tag{3.38}$$

Una formulación similar se usa para el perfil de velocidades cerca de paredes:

$$\mu_\tau^{vis} = \frac{U_1}{y^+};\ \mu_\tau^{log} = \frac{U_1}{\frac{1}{\kappa}ln(y^+) + C};\ \mu_\tau = \left[\left(\mu_\tau^{vis}\right)^4 + \left(\mu_\tau^{log}\right)^4\right]^{0,25} \tag{3.39}$$

3.3.6. Cálculo del tratamiento automático en paredes en OpenFOAM

```
        (...)
235     // Set omega and G
236     forAll(nutw, faceI)
237     {
```

```
238         label cellI = patch.faceCells()[faceI];
239
240         scalar w = cornerWeights[faceI];
241
242         scalar omegaVis = 6.0*nuw[faceI]/(beta1_*sqr(y[faceI]));
243
244         scalar omegaLog = sqrt(k[cellI])/(Cmu25*kappa_*y[faceI]);
245
246         omega[cellI] += w*sqrt(sqr(omegaVis) + sqr(omegaLog));
247
248         G[cellI] +=
249             w
250            *(nutw[faceI] + nuw[faceI])
251            *magGradUw[faceI]
252            *Cmu25*sqrt(k[cellI])
253            /(kappa_*y[faceI]);
254     }
        (...)
```

Como se vio en la ecuación 3.29 la velocidad de fricción se puede poner como función de la energía turbulenta cinética k como:

$$u_\tau = C_\mu^{1/4} k^{1/2} = u_\tau(k)$$

Sustituyendo esta velocidad de fricción en función de k en la ecuación 3.37 para obtener la ω de la subcapa logarítmica llegamos a la siguiente ecuación:

$$\omega_{log} = \frac{1}{0{,}3\kappa} \frac{C_\mu^{1/4} k^{1/2}}{y} \tag{3.40}$$

Al ser $C_\mu = 0{,}09$ su raíz cuadrada vale $C_\mu^{1/2} = 0{,}3$, por lo que sustituímos el valor de 0.3 de la ecuación 3.40 por $C_\mu^{1/2}$, obteniendo

$$\omega_{log} = \frac{1}{C_\mu^{1/2}\kappa} \frac{C_\mu^{1/4} k^{1/2}}{y} \tag{3.41}$$

Simplificando 3.41 llegamos a la expresión final de ω en la subcapa logarítmica.

$$\omega_{log} = \frac{\sqrt{k}}{C_\mu^{1/4}\kappa y} \tag{3.42}$$

El valor de ω en la subcapa viscosa aparece como *omegaVis* donde:

- ω_{visc} es *omegaVis*
- ν es *nuw[faceI]*

- *beta1_* es 0.075
- y^2 es *sqr(y[faceI])*

Así mismo vemos que el valor de ω en la subcapa logarítmica aparece definida en el código de tal manera que:

- ω_{log} es *omegaLog*
- $k^{1/2}$ es *sqrt(k[cellI])*
- $C_\mu^{1/4}$ es *Cmu25*
- y es *y[faceI]*

3.3.7. Código de la función de pared estándar en OpenFOAM

```
// Set epsilon and G
  232     forAll(nutw, facei)
  233     {
  234         label celli = patch.faceCells()[facei];
  235
  236         scalar w = cornerWeights[facei];
  237
  238         epsilon[celli] += w*Cmu75*pow(k[celli], 1.5)/(kappa_*y[facei]);
  239
  240         G[celli] +=
  241             w
  242            *(nutw[facei] + nuw[facei])
  243            *magGradUw[facei]
  244            *Cmu25*sqrt(k[celli])
  245            /(kappa_*y[facei]);
  246     }
```

En la línea 238 de la función *epsilonLowReWallFunctionFvPatchScalarField.C* vemos cómo se implementa la ecuación 3.28, donde:

- ϵ es *epsilon[celli]*
- $C_\mu^{3/4}$ es *Cmu75*
- $k_p^{3/2}$ es *pow(k[celli], 1.5)*
- κy es *kappa_*y[facei]*

$$\epsilon = \frac{C_\mu^{3/4} k_p^{3/2}}{\kappa y}$$

Capítulo 4

Resultados

4.1. Definición del problema

Se pretende realizar un estudio sobre el comportamiento fluidodinámico de un cilindro de longitud infinita (*problema 2D*) al ser sumergido en una corriente a distintos números de Reynolds. La figura 4.1 refleja las dimensiones geométricas que se manejan en el actual estudio. Los parámetros del estudio son:

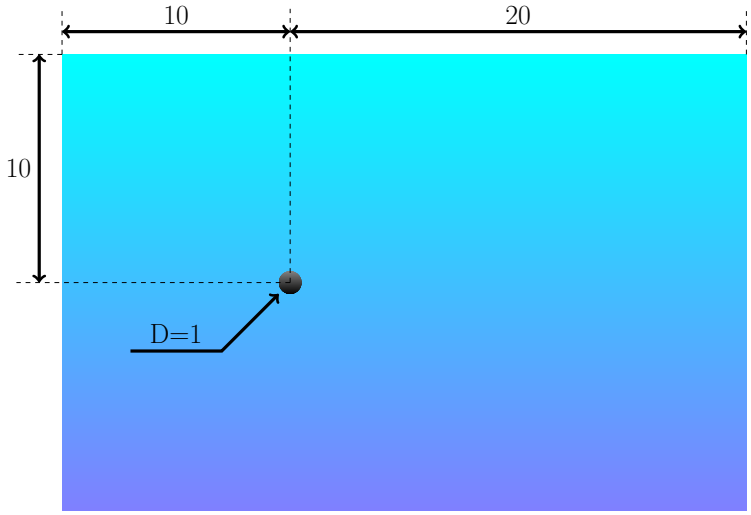


Figura 4.1: Dominio computacional. Unidades en [m].

- Viscosidad cinemática: $\nu = 10^{-6}$ $[m^2/s]$.
- Diámetro del cilindro: $\varnothing = 1$ $[m]$.
- Números de Reynolds usados: $Re = 10^4$; $Re = 5 \cdot 10^4$; $Re = 10^5$; $Re = 10^6$.
- Densidad del fluido: $\rho = 10^3$ $[kg/m^3]$.

4.2. Características del dominio computacional y de la discretización

4.2.1. Mallados

Los mallados usados son de tipo hexahédrico estructurado, con distinta cantidad de celdas según el estudio a realizar. Los mallados se dividen en 12 zonas representadas en la figura 4.2, donde zonas del mismo color indica que éstas tienen el mismo número de celdas. El círculo interior de color blanco es el cilindro.

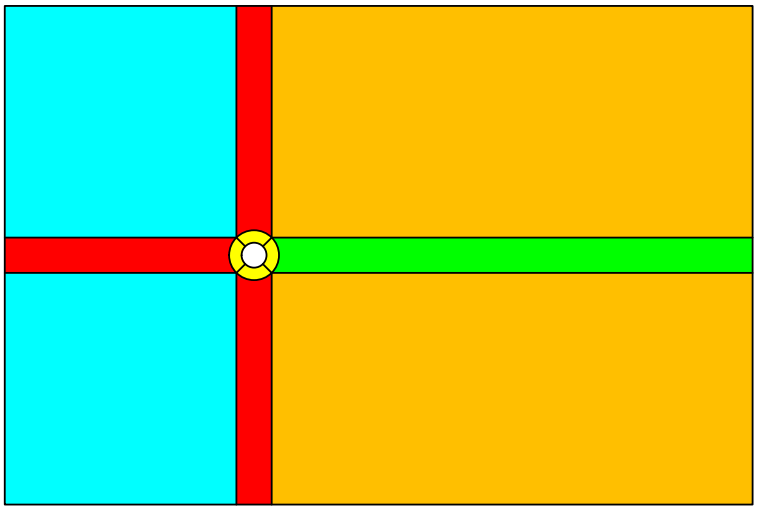


Figura 4.2: Descomposición del mallado. Las zonas del mismo color presentan las mismas cantidades de celdas.

Las figuras 4.4, 4.3 y 4.5 dan muestra del detalle de mallado entorno al cilindro según el números de Reynolds, y la tabla 4.1 visualiza las características de cada mallado. Se puede observar cómo a menor Reynolds las celdas entorno al cilindro son más esbeltas. Esto se debe a que a al disminuir en número de Reynolds las subcapas de la capa límite tienden a expandirse provocando que la zona logarítmica se encuentre muy alejada de la superficie, y puesto que es en el centroide de estas celdas quien debe caer en zona logarítmica da como resultado unas celdas esbeltas.

Malla	**Puntos**	**Caras**	**Caras internas**	**Celdas**
Malla 1	270040	537710	267670	134230
Malla 2	482784	962400	479616	240336
Malla 3	273736	545102	271366	136078

Tabla 4.1: Características de cada malla

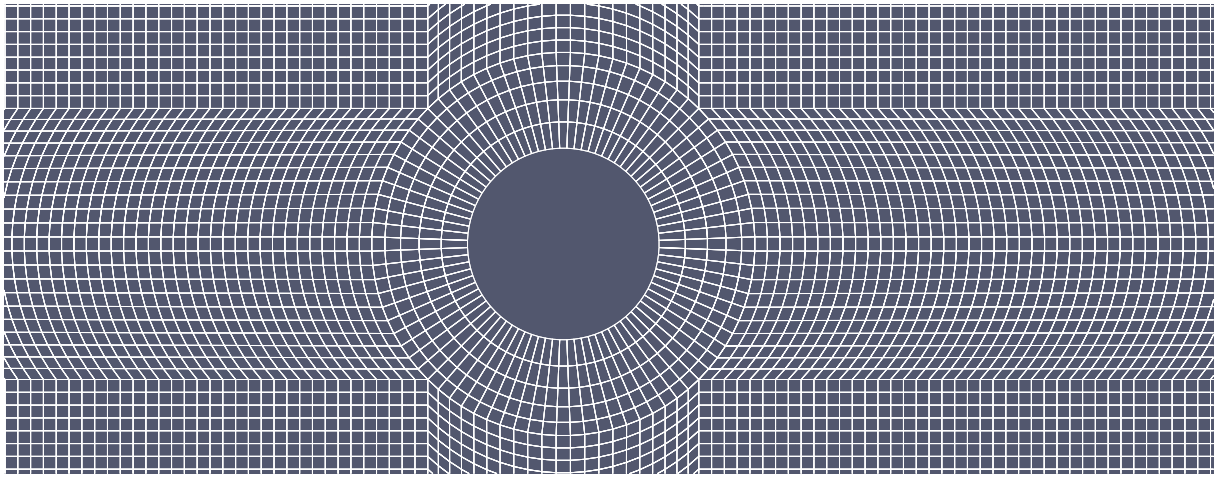


Figura 4.3: Malla 1 para $Re = 10^4$

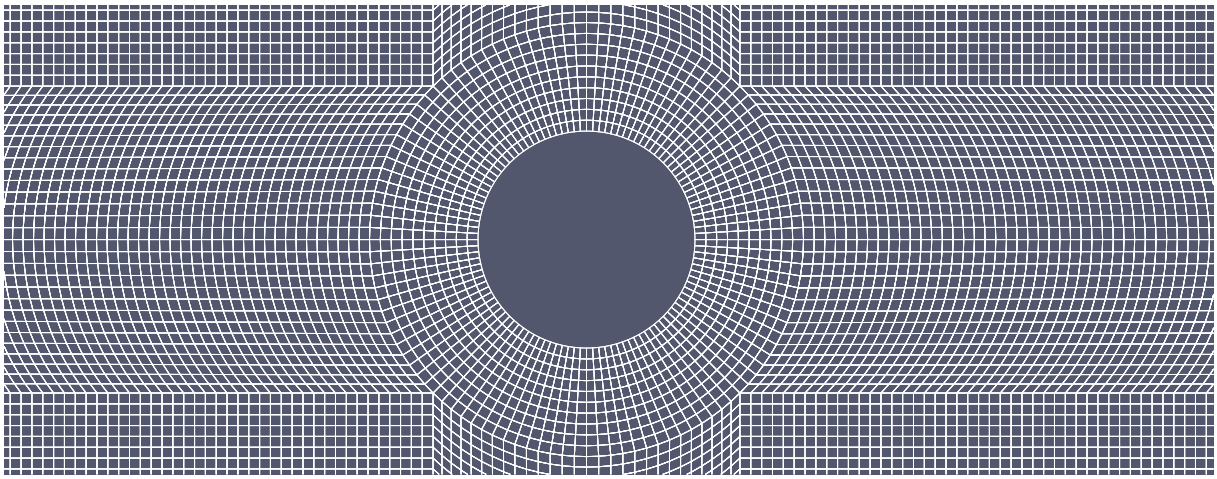


Figura 4.4: Malla 2 para $Re = 5 \cdot 10^4$ y $Re = 10^5$.

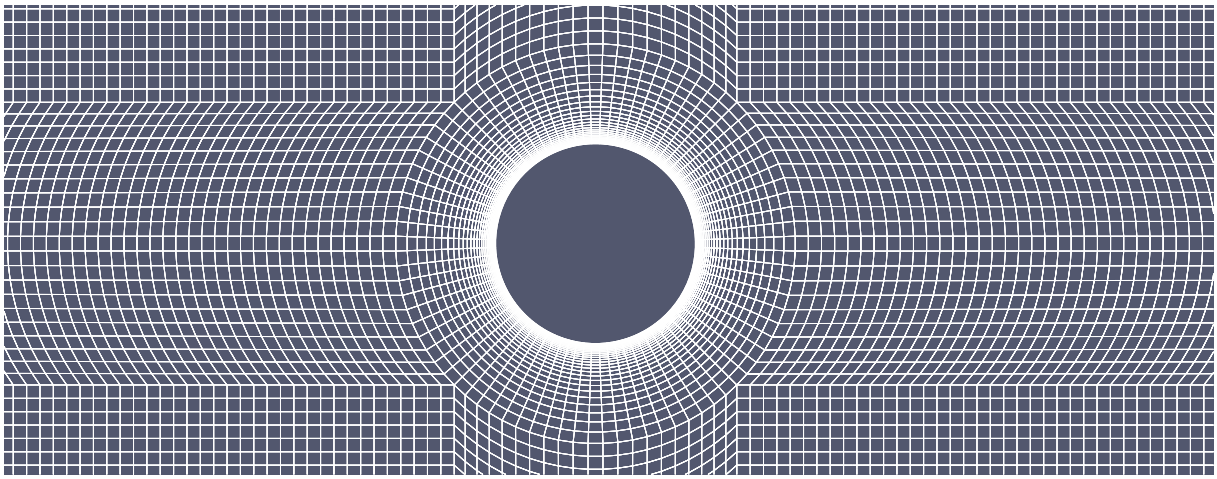


Figura 4.5: Malla 3 para $Re = 10^6$.

4.2.2. Condiciones de contorno e iniciales

En la figura 4.6 podemos ver los nombres que se le ha puesto en los contornos del dominio.

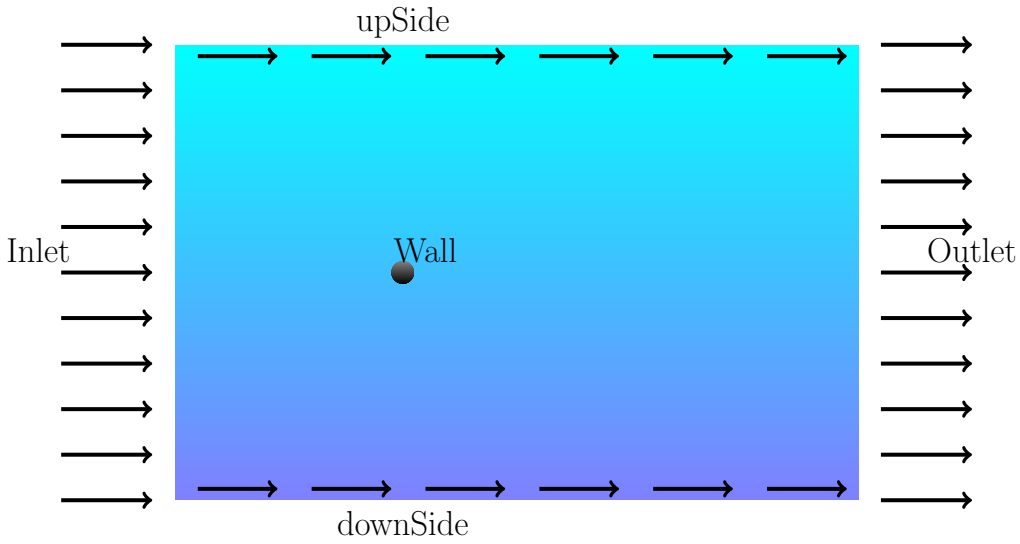


Figura 4.6: Nombre de los contornos.

Las condiciones de contorno para cada variable son las siguientes:

4.2.3. Presiones

- *Inlet*: Gradiente cero.
- *Outlet*: Valor fijo P=0.
- *upSide*: Gradiente cero.
- *downSide*: Gradiente cero.
- *Wall*: Gradiente cero.
- *Dominio*: Valor fijo P=0.

4.2.4. Velocidades

- *Inlet*: Valor fijo (distinto en cada simulación).
- *Outlet*: Gradiente cero.
- *upSide*: Superficie deslizante.
- *downSide*: Superficie deslizante.
- *Wall*: Valor fijo U=0.

- *Dominio*: Valor fijo U=0.

4.2.5. k

- *Inlet*: Valor fijo (distinto en cada simulación).
- *Outlet*: Gradiente cero.
- *upSide*: Gradiente cero.
- *downSide*: Gradiente cero.
- *Wall*: Wall Function para k.
- *Dominio*: Valor fijo (distinto en cada simulación).

4.2.6. epsilon

- *Inlet*: Valor fijo (distinto en cada simulación).
- *Outlet*: Gradiente cero.
- *upSide*: Gradiente cero.
- *downSide*: Gradiente cero.
- *Wall*: Wall Function para ϵ.
- *Dominio*: Valor fijo (distinto en cada simulación).

4.2.7. omega

- *Inlet*: Valor fijo (distinto en cada simulación).
- *Outlet*: Gradiente cero.
- *upSide*: Gradiente cero.
- *downSide*: Gradiente cero.
- *Wall*: Wall Function para ω.
- *Dominio*: Valor fijo (distinto en cada simulación).

4.2.8. viscosidad ν

- *Inlet*: Calculada por OpenFOAM.
- *Outlet*: Calculada por OpenFOAM.
- *upSide*: Calculada por OpenFOAM.
- *downSide*: Calculada por OpenFOAM.
- *Wall*: Wall Function para ν.
- *Dominio*: Valor fijo ν=0.

4.2.9. viscosidad $\nu\prime$

- *Inlet*: Gradiente cero.
- *Outlet*: Gradiente cero.
- *upSide*: Gradiente cero.
- *downSide*: Gradiente cero.
- *Wall*: Gradiente cero.
- *Dominio*: Valor fijo $\nu\prime$=0.

Los valores de epsilon, omega y velocidades en los contornos se calculan de la siguiente manera. Primero se impone un valor para la viscosidad, por ejemplo, $\nu = 10^{-6}\ [m^2/s]$, a continuación se puede calcular la velocidad haciendo uso del número de Reynolds, donde la longitud característica que se ha de tomar en este caso es el diámetro del cilindro, así,

$$Re = \frac{U_\infty \cdot L}{\nu} \longrightarrow U_\infty = \frac{Re \cdot \nu}{L}\ [m/s] \tag{4.1}$$

donde $L = \varnothing = 1\ [m]$.

La longitud de mezcla (*mixing length*) se impone un valor de $l = 7\,\%$ del diámetro, y la intensidad turbulenta se escoge un valor de $I = 2{,}5\,\%$, eligiendo en ambos casos los mismos valores que se recogen en el artículo de Stringer *et al* [2].

Con esto podemos ya calcular k, ϵ, ω.

$$k = \frac{3}{2}\left(U_\infty \cdot I\right)^2 \tag{4.2}$$

$$l = 0{,}07 \cdot L \tag{4.3}$$

$$\epsilon = \frac{C_\mu^{3/4} \cdot k^{3/2}}{l} \tag{4.4}$$

$$\omega = \frac{k^{3/2}}{k \cdot C_\mu \cdot l} \tag{4.5}$$

con $C_\mu = 0{,}09$.

De esta manera, los valores en los contornos y condiciones iniciales así calculados se muestran en la tabla 4.2.

$\boldsymbol{Re}$	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k}$	$9{,}38 \cdot 10^{-8}$	$2{,}34 \cdot 10^{-6}$	$9{,}38 \cdot 10^{-6}$	$9{,}38 \cdot 10^{-4}$
$\boldsymbol{\epsilon}$	$6{,}74 \cdot 10^{-11}$	$8{,}42 \cdot 10^{-9}$	$6{,}74 \cdot 10^{-8}$	$6{,}74 \cdot 10^{-5}$
$\boldsymbol{\omega}$	0,0486	0,243	0,486	4,86
$\mathbf{U}$	0,01	0,05	0,1	1

Tabla 4.2: Condiciones iniciales y de contorno. Unidades en el SI.

4.3. Definición de las magnitudes de estudio

Los coeficientes de fuerzas C_D y C_L son dos números adimensionales usados en ingeniería para estudiar las fuerzas de un cuerpo en el seno de un fluido. El coeficiente C_D es la adimensionalización de la fuerza de arrastre que en nuestro caso posee la misma dirección y sentido que el fluido, mientras que el C_L hace lo propio para la fuerza de sustentación. Así, para cualquier fuerza F, el coeficiente aerodinámico de dicha fuerza viene expresado por la ecuación 4.6.

$$C_F = \frac{F}{\frac{1}{2}\rho \cdot A \cdot U_\infty^2} \tag{4.6}$$

donde, ρ es la densidad del fluido, U_∞ la velocidad del fluido sin perturbar, y A es un área que, en el caso de un cilindro, se define como el diámetro multiplicado por la altura de éste, $A = \varnothing \cdot 1\ [m^2]$. Bien es cierto que se entiende que al ser un estudio bidimensional, esto implicaría que la geometría del cilindro en el tercer eje sería infinita. Sin embargo se toma como altura de cilindro la unidad, que además coincide con el espesor de la malla que se ha utilizado para resolver los distintos casos.

4.3.1. Coeficiente de resistencia

$$C_D = \frac{Drag}{\frac{1}{2}\rho \cdot A \cdot U_\infty^2} \tag{4.7}$$

donde

- *Drag* es la fuerza de arrastre, componente de la fuerza que ejerce el fluido sobre el cilindro con la misma dirección que la dirección principal del fluido.
- $\rho = 1000\ [kg/m^3]$.
- $A = 1\ [m^2]$.
- $U_\infty = 0{,}01\ [m/s]$ si $Re = 10^4$; $U_\infty = 0{,}05\ [m/s]$ si $Re = 5 \cdot 10^4$; $U_\infty = 0{,}1\ [m/s]$ si $Re = 10^5$; y $U_\infty = 1\ [m/s]$ si $Re = 10^6$.

4.3.2. Coeficiente de sustentación

$$C_L = \frac{Lift}{\frac{1}{2}\rho \cdot A \cdot U_\infty^2} \tag{4.8}$$

donde

- *Lift* es la fuerza de sustentación, componente de la fuerza que ejerce el fluido pero con la dirección perpendicular a la dirección principal del fluido.
- $\rho = 1000\ [kg/m^3]$.
- $A = 1\ [m^2]$.
- $U_\infty = 0{,}01\ [m/s]$ si $Re = 10^4$; $U_\infty = 0{,}05\ [m/s]$ si $Re = 5 \cdot 10^4$; $U_\infty = 0{,}1\ [m/s]$ si $Re = 10^5$; y $U_\infty = 1\ [m/s]$ si $Re = 10^6$.

4.3.3. Número de Strouhal

El número de Strouhal es un número adimensional usado para medir la frecuencia de desprendimiento de vórtices en un ciclo completo, en otras palabras, es la adimensionalización de una frecuencia, y esto es,

$$Str = \frac{f \cdot L}{U_\infty} \tag{4.9}$$

donde f es la frecuencia de desprendimiento de vórtices en *[Hz]*. Esta frecuencia se mide para un ciclo completo. En el caso de que nos ocupa, un ciclo de desprendimiento de vórtices comprende desde la aparición de un vórtice, por ejemplo, el de la parte superior, para que a continuación se desprenda el vórtice del lado opuesto, hasta volver a desprenderse un tercer vórtice en la parte superior, cerrando de esta manera el primer ciclo de desprendimiento de vórtices. La gráfica que refleja uno de estos ciclos es la gráfica del C_L frente al tiempo, donde los máximos y mínimos representan desprendimientos de vórtices en una y otro extremo del cilindro. La frecuencia del C_D es el doble al presentar un máximo cuando se desprende un vórtice independientemente de en qué lado se produzca éste. Esto se ve mejor reflejado en la figura 4.7.

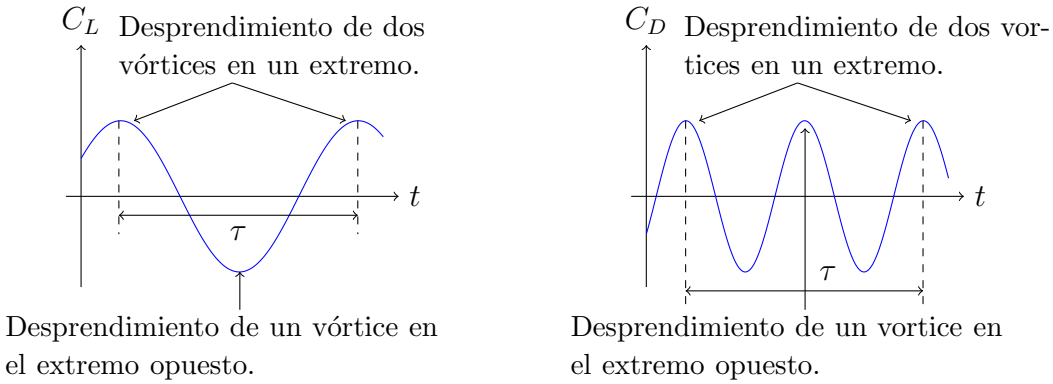


Figura 4.7: Ciclo completo de desprendimiento de vórtices.

4.4. Parámetros que caracterizan el efecto de los flujos transversales

Una vez explicadas las magnitudes de estudio procedemos a mostrar los resultados obtenidos tras las simulaciones. Se exponen a continuación los coeficientes aerodinámicos medidos sobre el cilindro y se comparan con los valores obtenidos por otros autores como Zdravkovich [14], Norberg [15] y Achenbach y Heinecke [16] que aparecen recopilados por R.M. Stringer *et al* en [2].

4.4.1. Coeficiente de resistencia

La tabla 4.3 muestra los coeficientes de resistencias tomados alrededor del cilindro obtenidos en las simulaciones con OpenFOAM y se comparan con los resultados obtenidos por Zdravkovich [14] en la tabla 4.4. La figura 4.8 compara los resultados con los obtenidos a partir de [2].

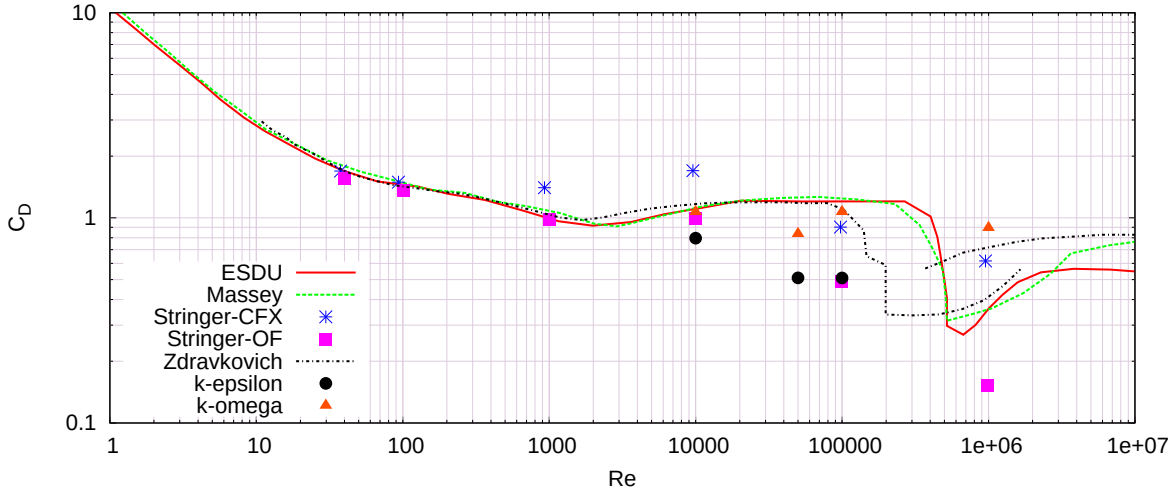


Figura 4.8: Cd frente Re.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k-\epsilon}$	0,796	0,510	0,509	
$\mathbf{k\,\omega-SST}$	1,07	0,836	0,690	0,865
Zdravkovich	1,1662	1,1836	1,1077	$0,4 \sim 0,7$

Tabla 4.3: Resultados del coeficientes de arrastre obtenidos a diversos Reynolds para los modelos de turbulencia $k-\epsilon$ y $k\,\omega-$SST .

Y así, los errores cometidos tomando como referencia los datos de Zdravkovich [14] son:

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k-\epsilon}$	31,744 %	56,911 %	54,049 %	
$\mathbf{k\,\omega-SST}$	8,2490 %	29,368 %	37,709 %	116,25 % $\sim$ 23,571 %

Tabla 4.4: Porcentaje de error cometido en las simulaciones comparándolo con los datos aportados por Zdravkovich [14].

4.4.2. Coeficiente de sustentación

Ahora vemos la media cuadrática los coeficientes de sustentación en la tabla 4.5 y los errores cometidos en las simulaciones por cada modelo de turbulencia con respecto a los datos obtenidos por Norberg [15]. La figura 4.9 compara los resultados con los recopilados en [2].

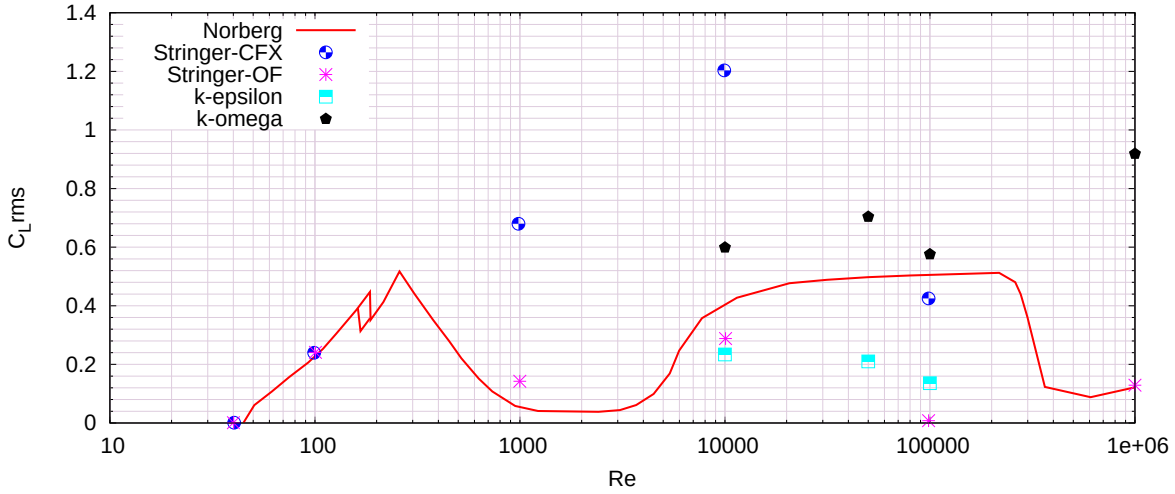


Figura 4.9: Clrms frente Re.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k-\epsilon}$	0,234	0,121	0,136	
$\mathbf{k\,\omega-SST}$	0,599	0,704	0,576	0,6318
Norberg	0,40038	0,4975	0,50465	0,1216

Tabla 4.5: Resultados de la media cuadrática del coeficiente de sustentación obtenidos a diversos Reynolds para los modelos de turbulencia $k-\epsilon$ y $k\,\omega$−SST, junto con los datos aportados por Norberg en [15] y recopilados en [2]

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k-\epsilon}$	41,556 %	75,678 %	73,051 %	
$\mathbf{k\,\omega-SST}$	49,608 %	41,508 %	14,139 %	419,57 %

Tabla 4.6: Porcentaje de error cometidos en la medición de la media cuadrática del coeficiente de sustentación comparado con los datos aportados por Norberg [15] y recopilados en [2].

4.4.3. Número de Strouhal

La tabla 4.7 muestra los valores obtenidos en la medición del número de Strouhal para los distintos Reynolds en los que se ha practicado la simulación. La tabla 4.8 muestra los errores cometidos en la medición del número de Strouhal al compararlo con los valores obtenidos por Achenbach y Heinecke [16] y por Norberg [15] sacados del artículo [2].

En la gráfica 4.10 se compara los resultados conseguidos aquí con los obtenidos por otros autores y publicados en [2].

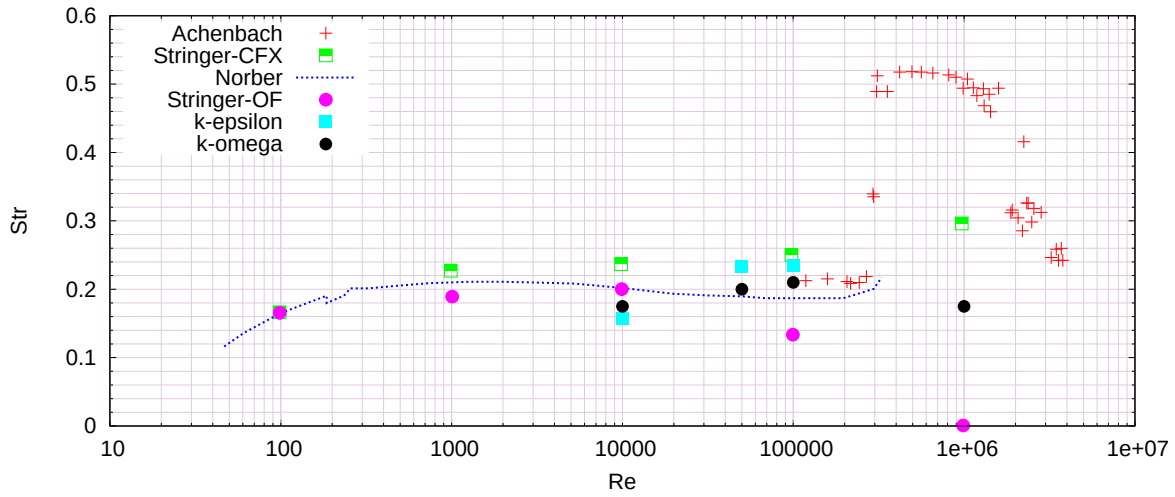


Figura 4.10: Str frente Re.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k-\epsilon}$	0,1575	0,2336	0,2347	
$\mathbf{k\,\omega-SST}$	0,1754	0,1992	0,2101	0,1736
Norberg	0,20186	0,18953	0,18684	
Achenbach y Heinecke				0,5

Tabla 4.7: Resultados de la medición del número de Strouhal obtenidos a diversos Reynolds para los modelos de turbulencia $k-\epsilon$ y $k\,\omega$−SST.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{k-\epsilon}$	21,976 %	23,252 %	25,615 %	
$\mathbf{k\,\omega-SST}$	13,108 %	5,1021 %	12,449 %	65,280 %

Tabla 4.8: Porcentaje de error de la medición del número de Strouhal obtenidos a diversos Reynolds para los modelos de turbulencia $k-\epsilon$ y $k\,\omega$−SST, comparado con los valores obtenidos por Achenbach y Heinecke [16] y por Norberg [15].

4.4.4. Valores del y^+

Las tablas 4.9 y 4.10 muestran los valores medios del y^+ medidos durante la ejecución de las simulaciones; el primero usando el modelo $k-\epsilon$ y el segundo usando el modelo $k\,\omega$−SST.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$
$\mathbf{y^+_{min}}$	11,3058	14,5724	29,2568
$\mathbf{y^+_{max}}$	64,284	120,953	230,379
$\mathbf{y^+_{med}}$	43,4938	74,3594	142,13

Tabla 4.9: Valores obtenidos del y^+ para el modelo $k - \epsilon$.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$	$\mathbf{10^6}$
$\mathbf{y^+_{min}}$	7,72658	12,0316	24,4487	7,67761
$\mathbf{y^+_{max}}$	35,3342	77,8201	153,689	113,601
$\mathbf{y^+_{med}}$	27,0922	48,6111	92,3227	57,5029

Tabla 4.10: Valores obtenidos del y^+ para el modelo $k\,\omega-$SST.

4.4.5. Patrones de flujo

En este apartado se presentará los campos de presiones obtenidos de las distintas simulaciones realizadas durante el estudio. Se puede observar en ellos cómo se desprenden los vórtices justo detrás del obstáculo, y cómo al aumentar el número de Reynolds los vórtices se encuentran más juntos.

4.4.5.1. Campos de velocidades obtenidos

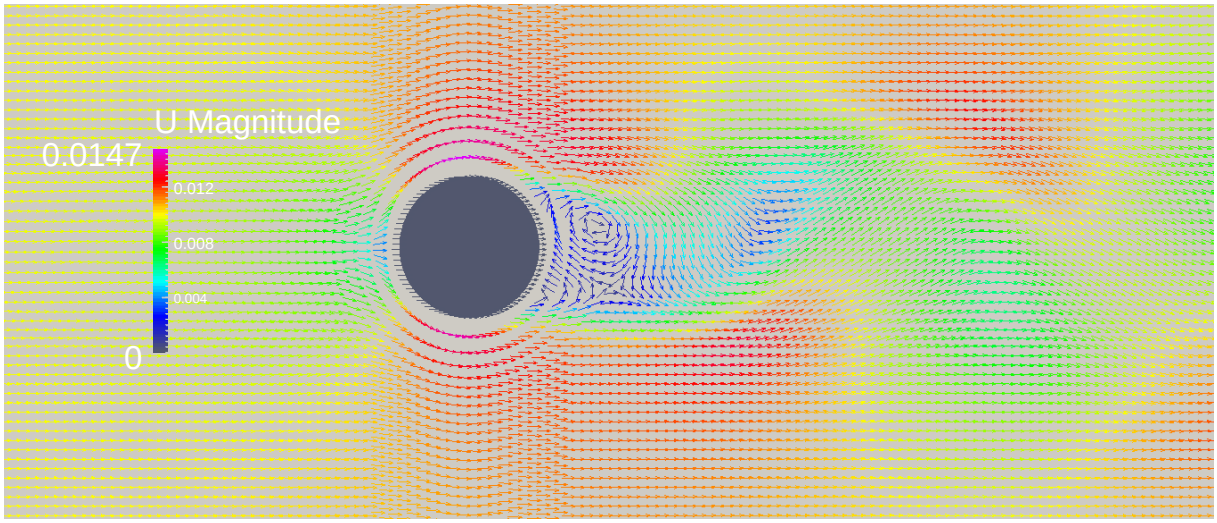


Figura 4.11: Campo de velocidades para $Re = 10^4$ con el modelo $k - \epsilon$.

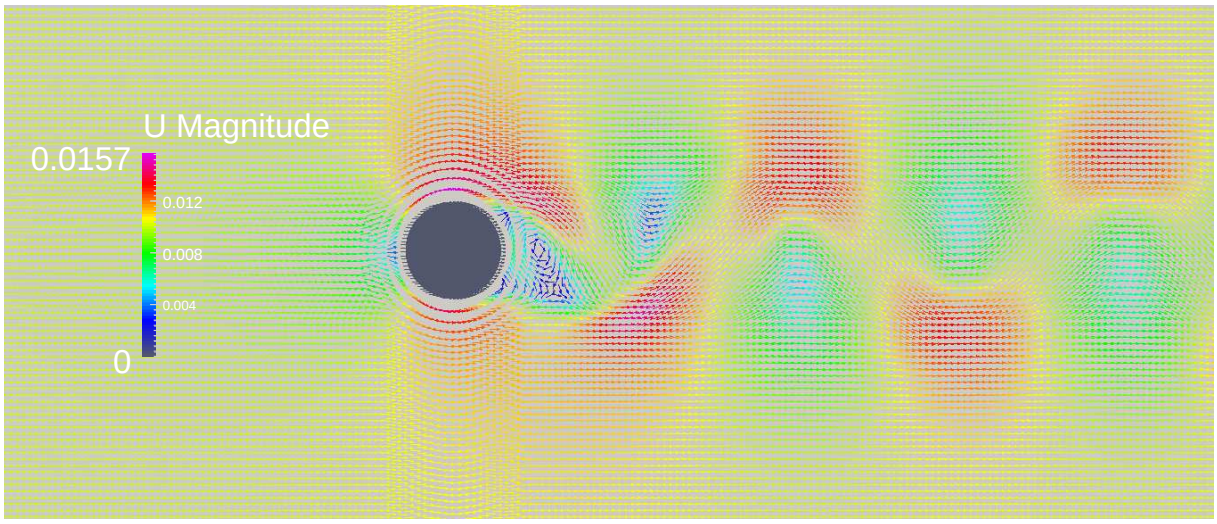


Figura 4.12: Campo de velocidades para $Re = 10^4$ con el modelo $k - \omega$.

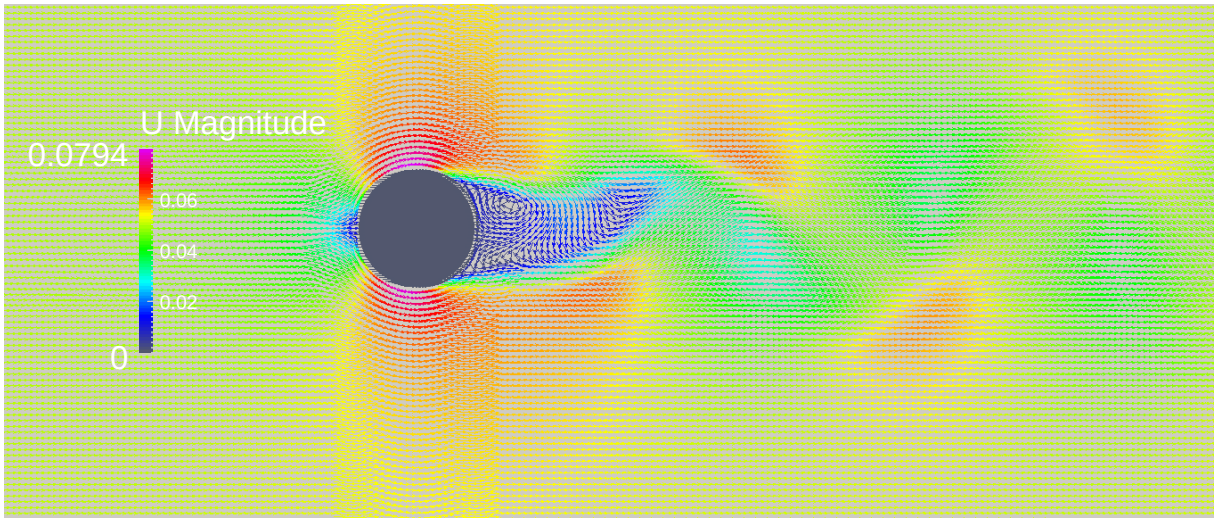


Figura 4.13: Campo de velocidades para $Re = 5 \cdot 10^4$ con el modelo $k - \epsilon$.

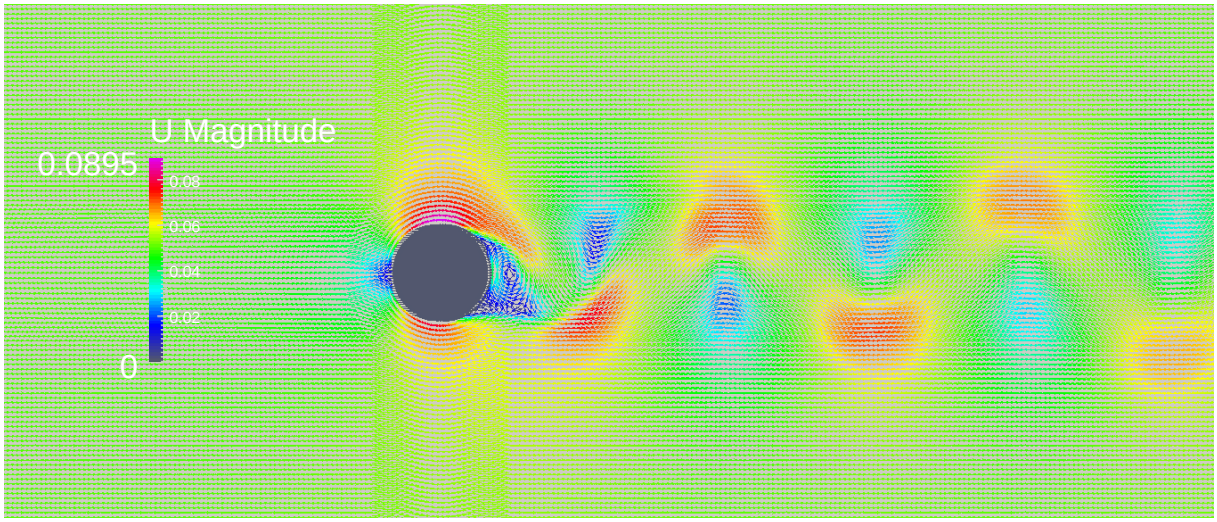


Figura 4.14: Campo de velocidades para $Re = 5 \cdot 10^4$ con el modelo $k - \omega$.

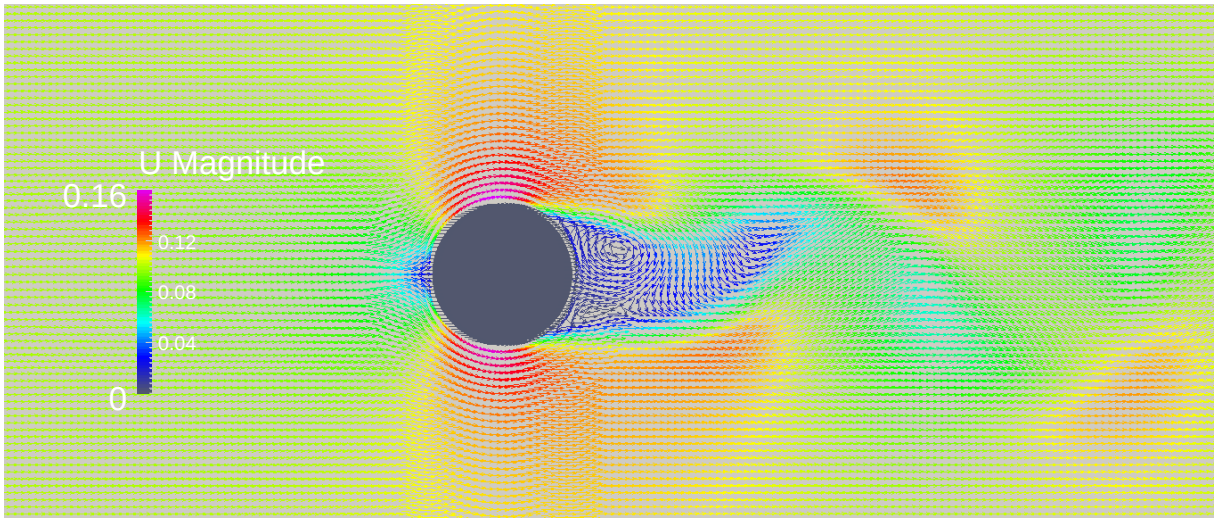


Figura 4.15: Campo de velocidades para $Re = 10^5$ con el modelo $k - \epsilon$.

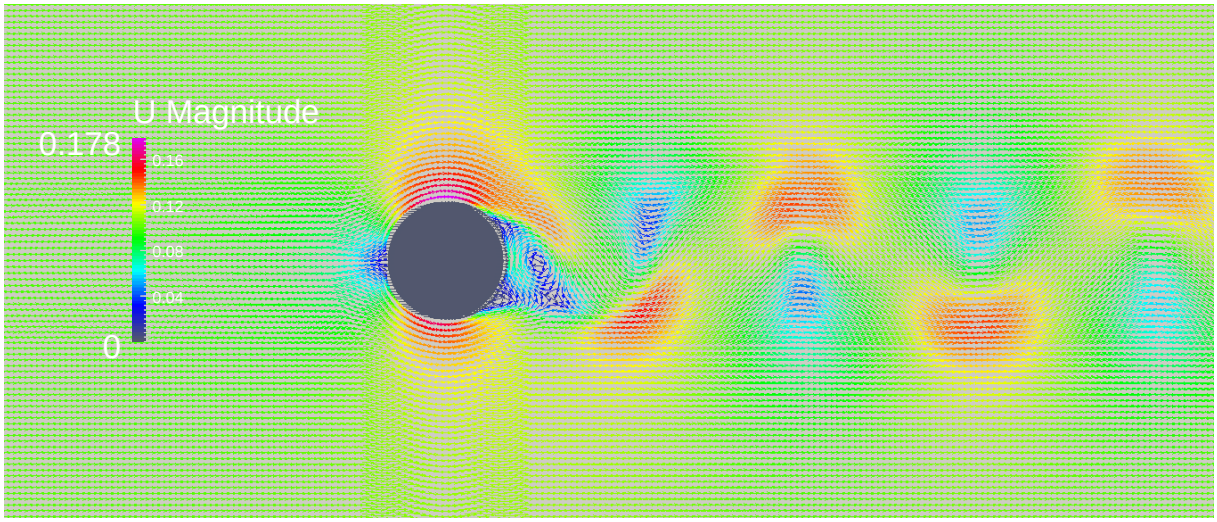


Figura 4.16: Campo de velocidades para $Re = 10^5$ con el modelo $k - \omega$.

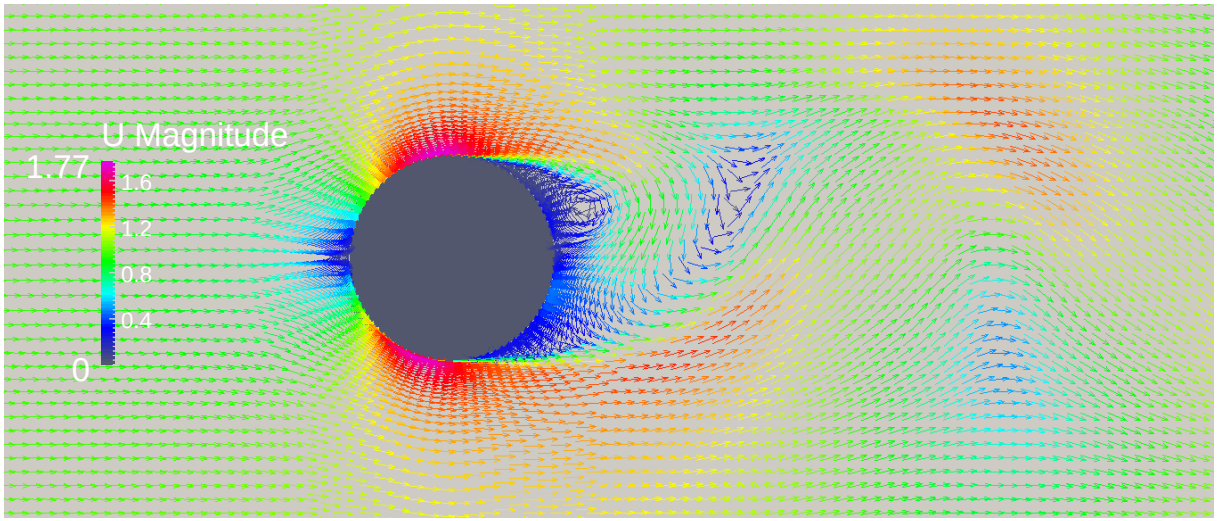


Figura 4.17: Campo de velocidades para $Re = 10^6$ con el modelo $k - \omega$.

4.4.5.2. Campos de presiones obtenidos

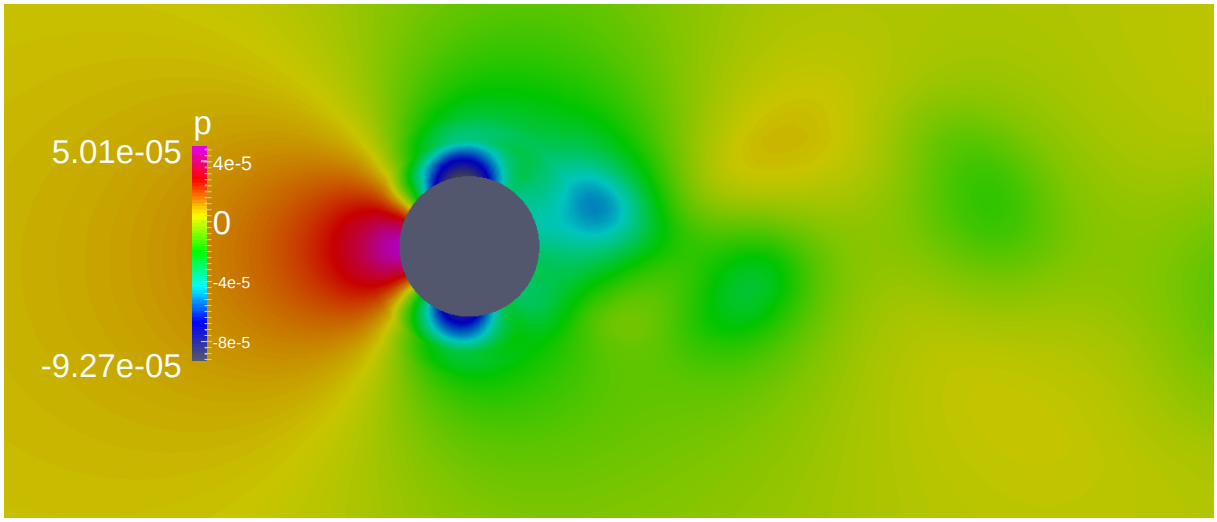


Figura 4.18: Campo de presiones para $Re = 10^4$ con el modelo $k - \epsilon$.

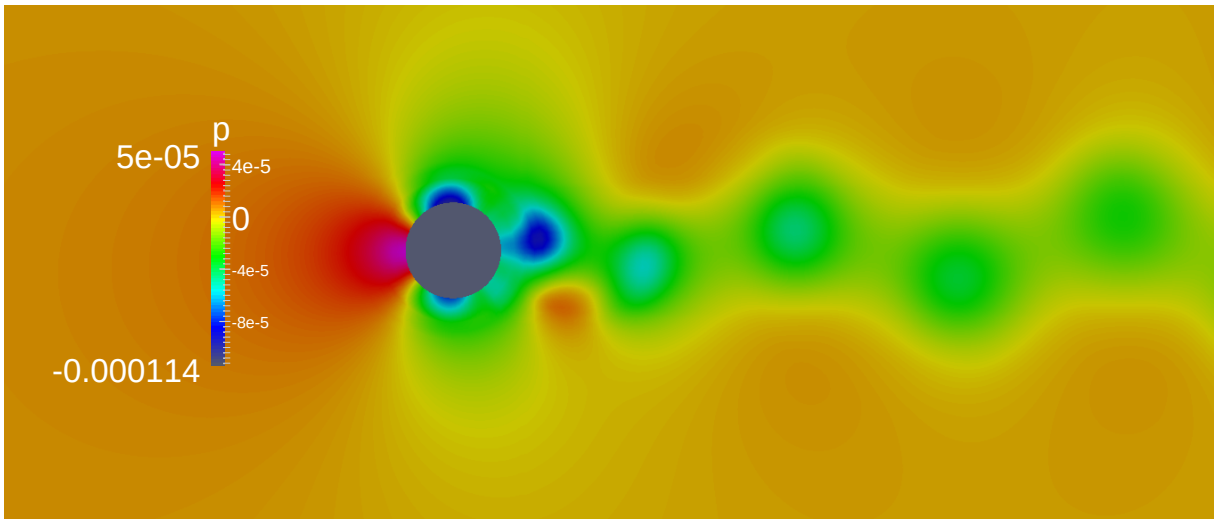


Figura 4.19: Campo de presiones para $Re = 10^4$ con el modelo $k - \omega$.

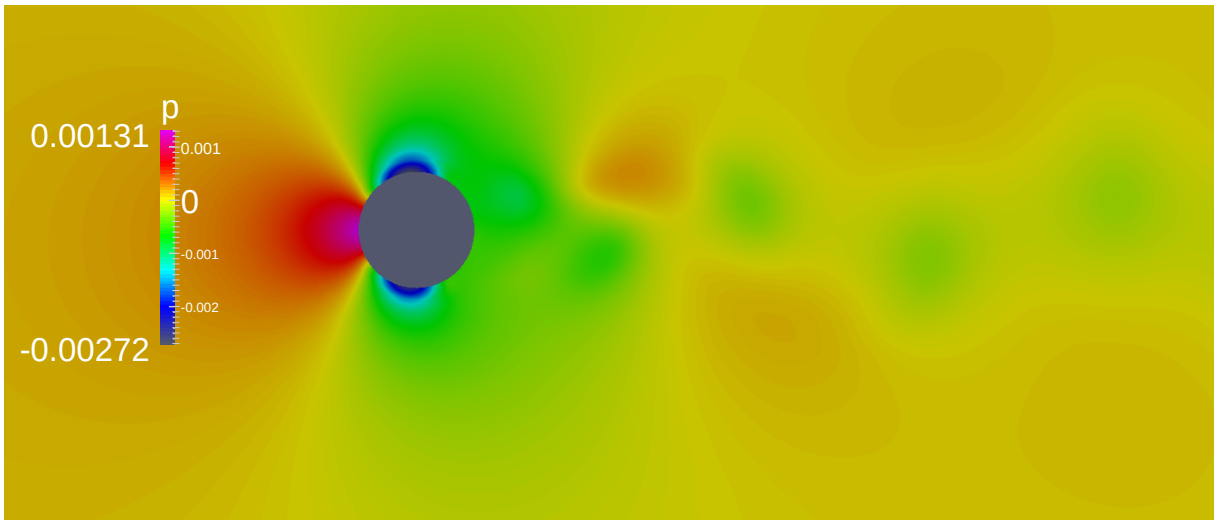


Figura 4.20: Campo de presiones para $Re = 5 \cdot 10^4$ con el modelo $k - \epsilon$.

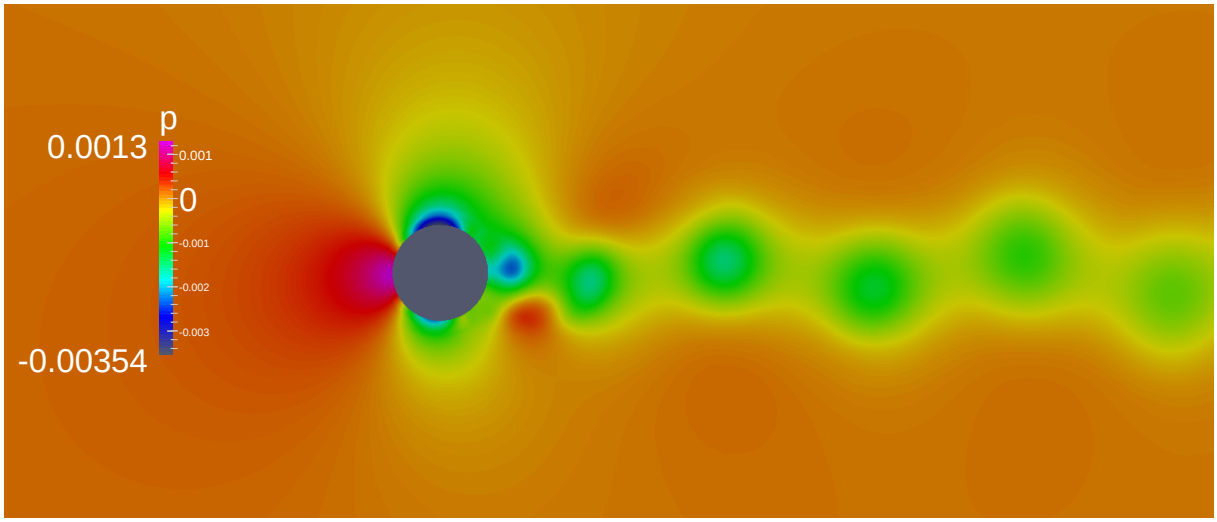


Figura 4.21: Campo de presiones para $Re = 5 \cdot 10^4$ con el modelo $k - \omega$.

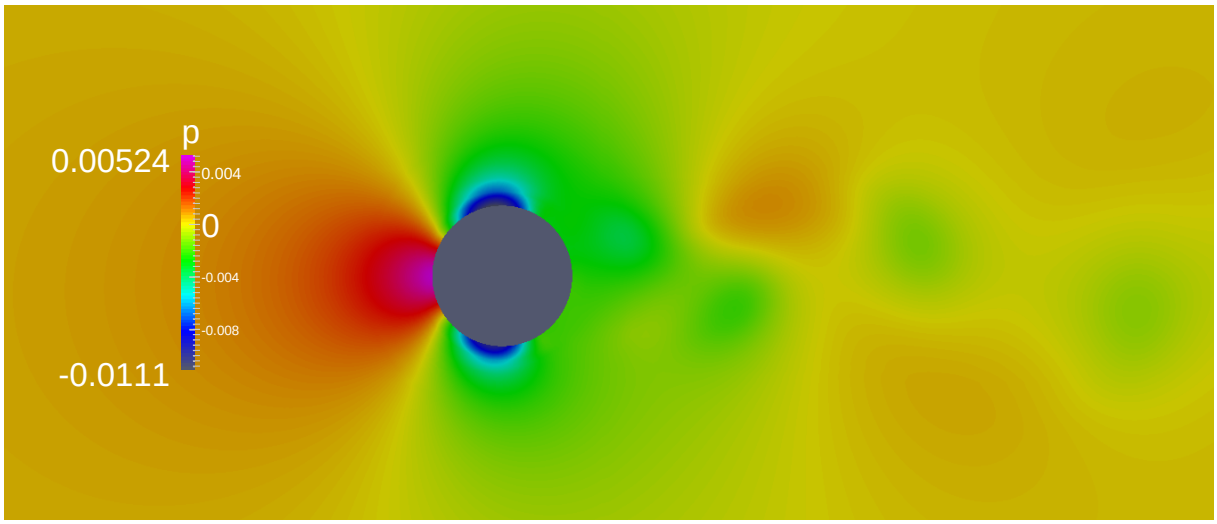


Figura 4.22: Campo de presiones para $Re = 10^5$ con el modelo $k - \epsilon$.

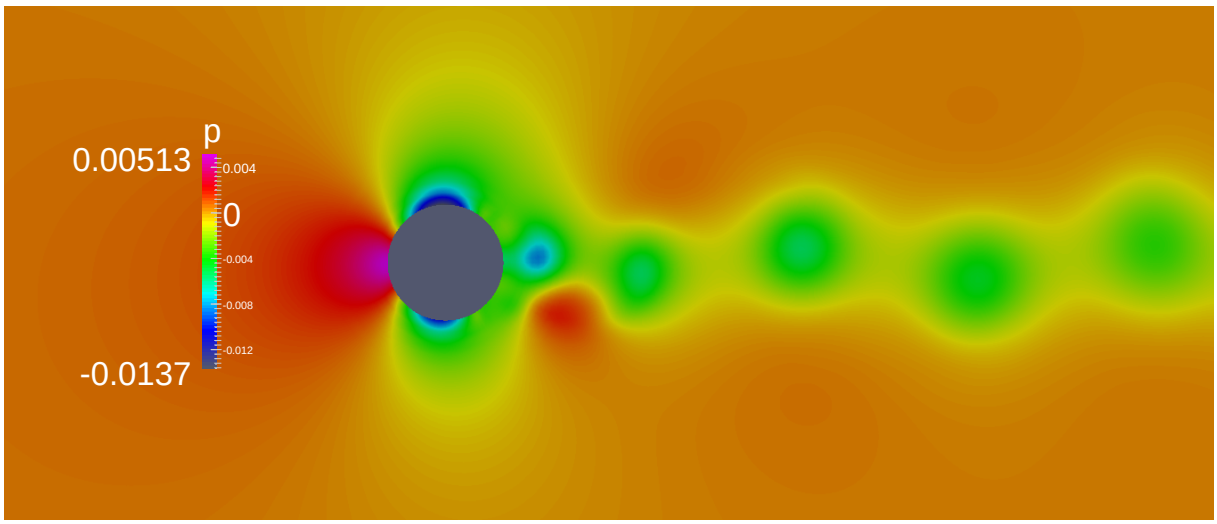


Figura 4.23: Campo de presiones para $Re = 10^5$ con el modelo $k - \omega$.

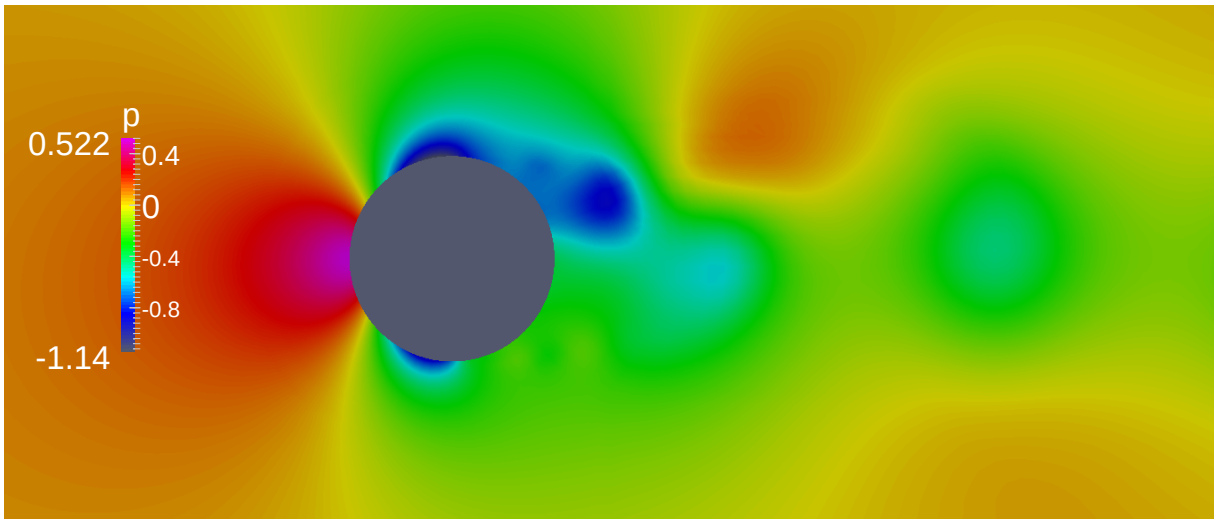


Figura 4.24: Campo de presiones para $Re = 10^6$ con el modelo $k - \omega$.

4.5. Estimación de la velocidad de fricción para el diseño de mallas

Con los resultados obtenidos se puede buscar una ley que nos permita, de manera aproximada, obtener una relación empírica entre el valor de la posición del primer centroide y la velocidad U_∞.

Recordando la definición de y^+ desde la ecuación 3.10,

$$y^+ = \frac{y \cdot u_\tau}{\nu}$$

donde y es la distancia normal del primer nodo a la pared, u_τ la velocidad tangencial y ν la viscosidad cinemática. Por tanto,

$$u_\tau = \frac{y^+ \cdot \nu}{y}$$

Una vez obtenidos los valores de u_τ imponemos la hipótesis de la existencia de una relación lineal entre la velocidad libre de perturbaciones y la velocidad tangencial u_τ. De esta manera tenemos que,

$$u_\tau = U_\infty \cdot C_x \tag{4.10}$$

En las gráficas (4.25) y (4.26) se observa que C_x sigue una parábola frente al número de Reynolds. Estas parábolas son:

- Modelo $k - \epsilon$.

$$\begin{aligned} C_{x,min} =& 0{,}035100 - 3{,}6505 \cdot 10^{-7} \cdot Re + 2{,}4454 \cdot 10^{-12} \cdot Re^2 \\ C_{x,max} =& 0{,}17517 + 5{,}5300 \cdot 10^{-7} \cdot Re - 4{,}8974 \cdot 10^{-12} \cdot Re^2 \\ C_{x,med} =& 0{,}12320 - 1{,}2930 \cdot 10^{-7} \cdot Re + 1{,}7000 \cdot 10^{-13} \cdot Re^2 \end{aligned} \tag{4.11}$$

- Modelo $k\,\omega$−SST.

$$\begin{aligned} C_{x,min} =& 0{,}0043869 + 1{,}9027 \cdot 10^{-6} \cdot Re - 1{,}7540 \cdot 10^{-11} \cdot Re^2 \\ C_{x,max} =& 0{,}089721 + 1{,}0022 \cdot 10^{-6} \cdot Re - 6{,}8863 \cdot 10^{-12} \cdot Re^2 \\ C_{x,med} =& 0{,}075270 + 7{,}8300 \cdot 10^{-8} \cdot Re - 1{,}0367 \cdot 10^{-12} \cdot Re^2 \end{aligned} \tag{4.12}$$

Fernández Oro [6] *(página 306)* aporta una fórmula empírica para obtener las velocidades de fricción en función del número de Reynolds tanto para flujos en conductos interno como para flujos externos. Esta expresión es

$$u_\tau = \overline{u_e}\sqrt{\overline{C_f}/2} \tag{4.13}$$

donde

- $\overline{u_e}$ es la velocidad del fluido libre de perturbaciones.
- $\overline{C_f}/2 \approx 0{,}037 Re_L^{-0,2}$ para placa plana, (flujo externo).
- $\overline{C_f}/2 \approx 0{,}039 Re_D^{-0,25}$ para conductos, (flujo interno).

En las gráficas 4.25 y 4.26 se compara los valores de C_x obtenidos en nuestro estudio con el valor obtenido de la ecuación 4.13 donde $C_x = \sqrt{\overline{C_f}/2}$. Se puede ver en dichas gráficas que el valor medio y^+_{med} que hemos obtenido se encuentra muy próximo al propuesto por Fernández Oro en [6], encontrando en el modelo $k\ \omega-$SST menores diferencias que en el modelo $k-\epsilon$. Estos valores y los errores con respecto al obtenido usando la expresión de Fernández Oro las vemos cuantificadas en las tablas 4.11 y 4.12 respectivamente.

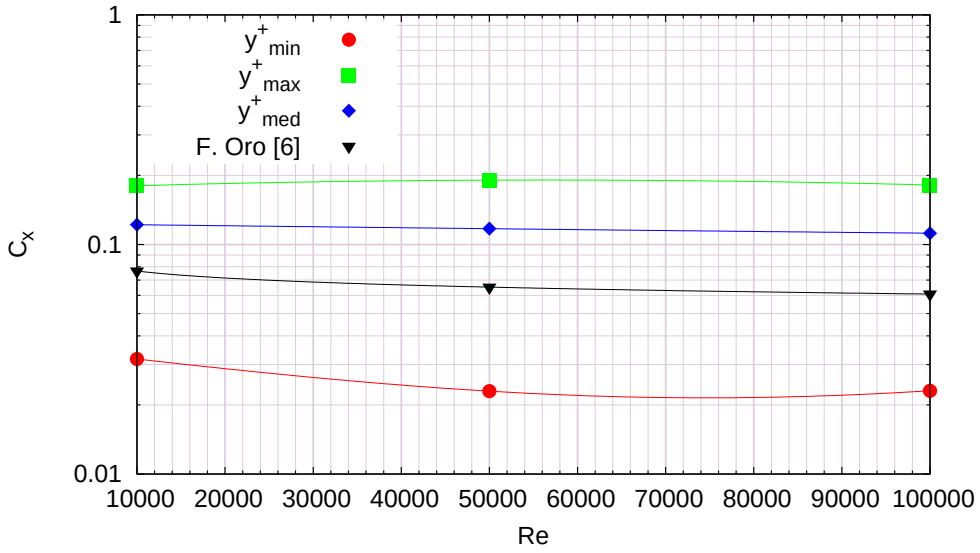


Figura 4.25: Valores de C_x a distintos Reynolds para el modelo $k-\epsilon$.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$
$\mathbf{C_{x,med}}$ **(k** $\boldsymbol{\omega}$ **− SST)**	0,075949	0,076593	0,072733
$\mathbf{C_{x,med}}$ **(k** $\boldsymbol{\epsilon}$**)**	0,121926	0,117162	0,111972
Fernández Oro	0,076577	0,065193	0,060828

Tabla 4.11: Valores de C_x para y^+_{med} obtenido de los modelos de turbulencia $k-\epsilon$ y $k\ \omega-$SST y el propuesto por Fernández Oro [6].

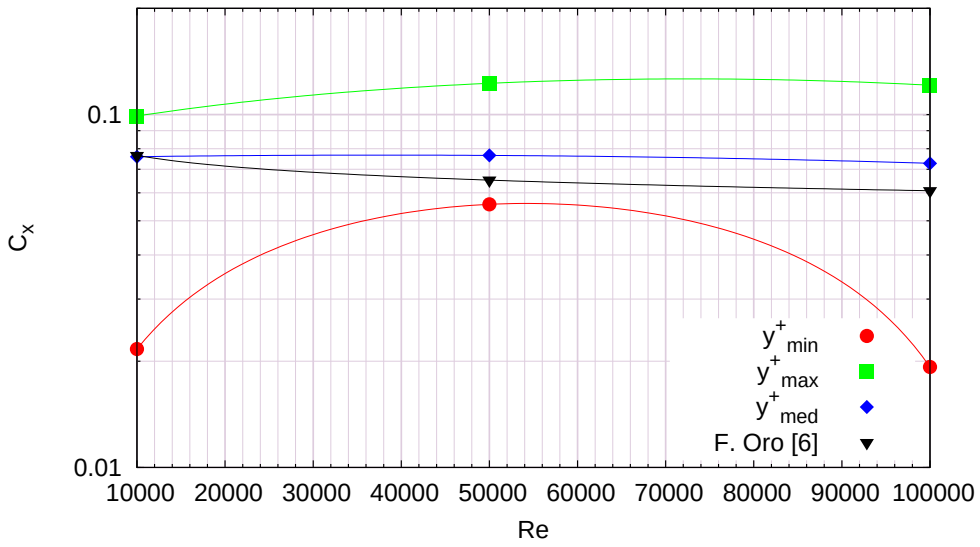


Figura 4.26: Valores de C_x a distintos Reynolds para el modelo $k\,\omega$−SST.

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$
$\mathbf{C_{x,med}}$ $\mathbf{(k\,\omega - SST)}$	0,82009 %	17,487 %	19,572 %
$\mathbf{C_{x,med}}$ $\mathbf{(k\,\epsilon)}$	59,220 %	79,716 %	84,080 %

Tabla 4.12: Errores de C_x para y^+_{med} obtenido de los modelos de turbulencia $k - \epsilon$ y $k\,\omega$−SST comparados con el propuesto por Fernández Oro [6].

Capítulo 5

Conclusiones y desarrollos futuros

Se ha realizado un estudio fluidodinámico bidimensional sobre un cilindro de longitud infinita, sumergido en una masa fluida de densidad constante a distintos números de Reynolds, $Re = 10^4$, $Re = 5 \cdot 10^4$, $Re = 10^5$, y $Re = 10^6$ (este último sólo $k\ \omega$−SST). El software utilizado ha sido el OpenFOAM, el cual basa sus esquemas numéricos en el método de los volúmenes finitos. Estos resultados han sido comparados con los resultados presentados por Stringer *et al* [2]. Como resultado final del presente estudio se llega a las siguientes conclusiones:

1. Los resultados obtenidos demuestran que el modelo de turbulencia $k\ \omega$−SST presenta mejores predicciones de los coeficientes de fuerzas y número de Strouhal notablemente mejor que usando el modelo $k-\epsilon$. Estas diferencias se hacen más evidentes a medida que se aumenta en número de Reynolds.

2. El modelo de turbulencia $k\ \omega$−SST presenta resultados mejores que los presentados por Stringer *et al* [2] hasta Reynolds=10^5 con errores:

Reynolds	$\mathbf{10^4}$	$\mathbf{5 \cdot 10^4}$	$\mathbf{10^5}$
$\mathbf{C_D}$	8,0855 %	28,544 %	1,9256 %
$\mathbf{C_L - rms}$	46,392 %	40,689 %	13,680 %
Str	13,275 %	5,3604 %	12,585 %

Tabla 5.1: Errores del modelo $k\ \omega$−SST al compararlos con Zdravkovich [14], recopilados en [2].

3. El tratamiento en paredes *Wall Function* ha demostrado ser una herramienta útil para aplicar condiciones de tipo Dirichlet principalmente, y de manera obligatoria, cuando se trabaja con el modelo $k-\epsilon$. El modelo $k\ \omega$−SST utiliza el tratamiento automático en paredes, si bien es más flexible que la *Wall Function* a la hora de colocar los nodos, tanto en subcapa viscosa como en la logarítmica, resulta útil poseer una herramienta que permita conocer en qué zona caen la mayoría de los nodos para así evitar que estos entren en zona de nadie (subcapa buffer). Por esto,

el proceso de simulación es un proceso que requiere de varias simulaciones previas que puedan ir prediciendo los tamaños de las distintas subcapas que conforman la capa límite, e ir remallando hasta conseguir una malla que sea adecuada a la geometría para un Reynolds dado.

4. A partir de $Re = 3 \cdot 10^5$ se observa un cambio de comportamiento del fluido reflejado principalmente por el número de Strouhal, que no ha podido ser reproducido por el OpenFOAM, ya que se obtiene valores muy similares para $Re = 10^6$ como para $Re = 10^5$. Estas diferencias pueden ser debidas a efectos tridimensionales que no son captados en dos dimensiones, o a que los modelos de turbulencia no son adecuados. Aún así es de destacar que los resultados obtenidos para $Re = 10^6$ resultan ser más aproximados a la realidad que los obtenidos por Stringer *et al* [2].

5. Así mismo, los resultados obtenidos para $Re = 10^5$ se asemejan en mayor medida a la realidad que los resultados numéricos obtenidos por Stringer *et al* [2].

6. Tal y como comentan Labbé y Wilson en [8], existen muchas discrepancias entre las simulaciones 2D y los resultados experimentales, por ello se propone como propuesta de trabajo futuro realizar simulaciones tridimensionales y comparar los resultados con los obtenidos experimentalmente por otros investigadores.

Referencias

[1] H. K. Versteeg y W. Malalasekera. **An introduction to computational fluid dynamics. The finite volume method**. Pearson Prentice Hall 02/16/2007. Segunda edición.

[2] R.M. Stringer, J. Zang y A.J. Hillis.**Unsteady RANS computations of flow around a circular cylinder for a wide range of Reynolds numbers**. Elsevier. Ocean Engineering 87 (2014) 1–9.

[3] Florian Menter y Thomas Esch. **Elements of industrial heat transfer predictions**. XVI congeresso brasileiro de engenharia mecânica. 16^{th} Brazilian congress of mechanical engineering. Cobem 2001, ABCM.

[4] Florian Menter, Jorge Carregal Ferreira, Thomas Esch, y Brad Konno. **The SST Turbulence Model with Improved Wall Treatment for Heat Transfer Predictions in Gas Turbines**. Proceedings of the International Gas Turbine Congress 2003 Tokyo. November 2-7, 2003. IGTC2003-TS-059. ANSYS CFX Germany. Staudenfeldweg 12, D-83624 Otterfing, GERMANY. CFX Asia Pacific K.K.

[5] Manuel José Chica González. **Estimación Computacional de la Resistencia al Avance de un Cuerpo Simple**. Máster Universitario en Tecnologías Industriales. Julio 2015. Universidad de Las Palmas de Gran Canaria. Escuela de Ingenierías Industriales y Civiles.

[6] Jesús Manuel Fernández Oro. **Técnicas numéricas en ingeniería de fluidos**. Reverté. 2012.

[7] C.H.K. Williamson. **Vortex dynamics in the cylinder wake**. Mechanical and Aerospace Engineering. . Annual Review of Fluid Mechanics, 28, pp. 477-539; 1996. Upson Hall. Cornell University, Ithaca, New York.

[8] D.F.L. Labbé y P.A. Wilson. **A numerical investigation of the effects of the spanwise length on the 3-D wake of a circular cylinder**. Elsevier. Journal of fluids and structures 23 (2007) 1168 – 1188. 6 de agosto del 2007.

[9] C. Lei, L. Cheng, K. Kavanagh. **Spanwise length effects on three-dimensional modelling of flow over a circular cylinder**. Elsevier. Computer methods in applied mechanics and engineering 190 (2001) 2909 – 2923.

[10] Héctor Rubén Díaz Ojeda. **Aplicación del método de volúmenes finitos al cálculo del parámetro de masa añadida de un cilindro vibrante en el seno de un fluido**. Universidad de Las Palmas de Gran Canaria. Máster Universitario en Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería. 2014-2015.

[11] www.cfdonline.com.

[12] www.openfoam.org.

[13] Eric Furbo. **Evaluation of RANS turbulence models for flow problems with significant impact of boundary layers**. Examensarbete 30 hp. December 2010. UPTEC F10061. UPPSALA UNIVERSITET. Handledare: Mattias Liefvendahl. Ämnesgranskare: Gunilla Kreiss. Examinator: Tomas Nyberg. ISSN: 1401-5757, UPTEC F10061.

[14] Zdravkovich, M.M., 1990. **Conceptual overview of laminar and turbulent flows past smooth and rough circular-cylinders.** J. Wind Eng. Ind. Aerodyn. 33, 53–62.

[15] Norberg, C., 2003. **Fluctuating lift on a circular cylinder: review and new measurements**. J. Fluids Struct. 17, 57–96.

[16] Achenbach, E., Heinecke, E., 1981. **On vortex shedding from smooth and rough cylinders in the range of reynolds-numbers** $6 \cdot 10^6$ to $5 \cdot 10^6$. J. Fluid Mech. 109, 239–251.

[17] Massey, B.S., 1989. Mechanics of Fluids. Van Nostrand Reinhold.

Apéndices

5.1. Fichero de datos para el caso $k - \epsilon$ a $Re = 10^4$

5.1.1. Capeta 0

5.1.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 6.7382e-11;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
```

```
type fixedValue;
value uniform 6.7382e-11;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 6.7382e-11;
    }
}


// ************************************************************************* //
```

5.1.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  9.375e-8;

boundaryField
```

```
{

    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            fixedValue;
value uniform 9.375e-8;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            kqRWallFunction;
value uniform 9.375e-8;
    }
}


// ************************************************************************* //
```

5.1.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
```

```
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    fontAndBack
    {
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.1.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}
```

```
// ************************************************************************* //
```

5.1.1.5. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
        type            zeroGradient;
    }

    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }
```

```
    fontAndBack
    {
type empty;
    }
}

// ************************************************************************* //
```

5.1.1.6. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.01 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.01 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }
```

```
    down
    {
type slip;
    }
// ************************************************************************* //
```

5.1.2. Carpeta constant

5.1.2.1. Archivo RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.1.2.2. Archivo transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
```

```
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.1.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //
```

5.1.3. Carpeta polyMesh

5.1.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])

define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Mathematicalconstants:
define(pi, 3.1415926536)

// Numeros de nodos
define(n1, 5)
define(n2, 21)
define(n3, 139)
define(n4, 139)
define(n5, 289)

// Radios
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))
```

```
define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))

// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)

// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 0.5)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


vertices
(
// Arco Derecho
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)

// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)
```

```
// Arco Izquierdo
//**************

// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******

(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)

(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
```

```
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)
```

```
// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//**************
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//**************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//**************
arc v5b v7b (-Rext 0 z0) // Arco exterior
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
```

```
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)

// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)
```

```
// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)

// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)

// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}

up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}
```

```
down
{
type patch;
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
);
```

5.1.4. Carpeta system

5.1.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       latestTime;

startTime       48000;

stopAt          endTime;

endTime         50000;

deltaT          1.25;

writeControl    timeStep;

writeInterval   400;
```

```
purgeWrite      2;

writeFormat     binary;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //

functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 1e-2;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.1.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
```

```
FoamFile
{
  version  2.0;
  format ascii;
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 16;

method simple;

simpleCoeffs
{
  n (4 4 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();

// ************************************************************************* //
```

5.1.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
```

```
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.1.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
```

```
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
```

```
}


// ************************************************************************* //

relaxationFactors
  {
      fields
      {
          p               0.3;
      }
      equations
      {
          U               0.7;
          k               0.7;
          epsilon         0.7;
          /*R              0.7;
          nuTilda         0.7; */
      }
  } /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
```

```
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //

relaxationFactors
  {
      fields
      {
          p               0.3;
      }
      equations
      {
          U               0.7;
          k               0.7;
          epsilon         0.7;
          /*R              0.7;
          nuTilda         0.7; */
      }
  }
```

5.2. Fichero de datos para el caso $k - \epsilon$ a $Re = 5 \cdot 10^4$

5.2.1. Capeta 0

5.2.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
```

```
|   \\  /    A nd           | Web:      www.OpenFOAM.org                    |
|    \\/     M anipulation  |                                               |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 8.4227e-09;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
type fixedValue;
value uniform 8.4227e-09;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 8.4227e-09;
    }
```

```
}


// ************************************************************************* //
```

5.2.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  2.3438e-06;

boundaryField
{

    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            fixedValue;
value uniform 2.3438e-06;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
```

```
    {
        type            zeroGradient;
    }

    hole
    {
        type            kqRWallFunction;
value uniform 2.3438e-06;
    }
}


// ************************************************************************* //
```

5.2.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    fontAndBack
    {
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
```

```
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.2.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
```

```
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

5.2.1.5. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;
```

```
boundaryField
{
   inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
        type            zeroGradient;
    }

    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }

     fontAndBack
     {
type empty;
     }
}

// ************************************************************************* //
```

5.2.1.6. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.05 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.05 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }

    down
    {
type slip;
    }
// ************************************************************************* //
```

5.2.2. Carpeta constant

5.2.2.1. Archivo RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.2.2.2. Archivo transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.2.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
```

```
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //
```

5.2.3. Carpeta polyMesh

5.2.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])

define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Mathematicalconstants:
define(pi, 3.1415926536)

// Numeros de nodos
define(n1, 10)
define(n2, 28)
define(n3, 186)
define(n4, 186)
define(n5, 386)

// Radios
```

```
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))

define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))

// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)

// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 1)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
vertices
(
// Arco Derecho
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)

// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)

// Arco Izquierdo
//**************

// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******

(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
```

```
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)

(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
```

```
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//**************
```

```
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//**************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//**************
arc v5b v7b (-Rext 0 z0) // Arco exterior
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
```

```
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)

// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)

// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)

// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)

// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
```

```
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}

up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}

down
{
type patch;
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
);
```

5.2.4. Carpeta system

5.2.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
```

```
|   \\  /    A nd           | Web:      www.OpenFOAM.org                    |
|    \\/     M anipulation  |                                               |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       latestTime;

startTime       8000;

stopAt          endTime;

endTime         10000;

deltaT          0.2;// Con 2.5 funciona

writeControl    timeStep;

writeInterval   500;

purgeWrite      2;

writeFormat     binary;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //

functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
```

```
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 5e-2;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.2.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version  2.0;
  format ascii;
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 10;

method simple;

simpleCoeffs
{
  n (2 5 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}
```

```
manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();

// ************************************************************************* //
```

5.2.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}
```

```
laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.2.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
```

```
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //

relaxationFactors
  {
      fields
      {
          p               0.15;
      }
      equations
      {
          U               0.85;
          k               0.7;
          epsilon         0.7;
          /*R              0.7;
          nuTilda         0.7; */
      }
  }
```

5.3. Fichero de datos para el caso $k - \epsilon$ a $Re = 10^5$

5.3.1. Capeta 0

5.3.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 6.7382e-08;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
type fixedValue;
value uniform 6.7382e-08;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }
```

```
    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 6.7382e-08;
    }
}


// ************************************************************************* //
```

5.3.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  9.3750e-06;

boundaryField
{

    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            fixedValue;
value uniform 9.3750e-06;
    }
```

```
    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            kqRWallFunction;
value uniform 9.3750e-06;
    }
}


// ************************************************************************* //
```

5.3.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    fontAndBack
    {
```

```
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.3.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

5.3.1.5. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
```

```
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
        type            zeroGradient;
    }

    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }

     fontAndBack
     {
type empty;
     }
}

// ************************************************************************* //
```

5.3.1.6. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
```

```
|   \\  /    A nd           | Web:      www.OpenFOAM.org                    |
|    \\/     M anipulation  |                                               |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.1 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.1 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }

    down
    {
type slip;
    }
// ************************************************************************* //
```

5.3.2. Carpeta constant

5.3.2.1. Archivo RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.3.2.2. Archivo transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.3.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //
```

5.3.3. Carpeta polyMesh

5.3.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])

define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
// Mathematicalconstants:
define(pi, 3.1415926536)

// Numeros de nodos
define(n1, 10)
define(n2, 28)
define(n3, 186)
define(n4, 186)
define(n5, 386)

// Radios
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))

define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))

// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)
```

```
// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 1)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


vertices
(
// Arco Derecho
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)

// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)

// Arco Izquierdo
//**************

// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******
```

```
(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)

(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)
```

```
// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)
```

```
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//**************
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//**************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//**************
arc v5b v7b (-Rext 0 z0) // Arco exterior
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
```

```
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)

// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)

// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)

// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)
```

```
// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}

up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}

down
{
type patch;
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
```

```
);
```

5.3.4. Carpeta system

5.3.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       latestTime;

startTime       10000;

stopAt          endTime;

endTime         11000;

deltaT          0.1;

writeControl    timeStep;

writeInterval   500;

purgeWrite      2;

writeFormat     binary;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;
```

```
// ************************************************************************* //

functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 1e-1;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.3.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version  2.0;
  format ascii;
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 10;

method simple;

simpleCoeffs
```

```
{
  n (2 5 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();

// ************************************************************************* //
```

5.3.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
```

```
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.3.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //

relaxationFactors
  {
      fields
      {
          p               0.15;
      }
      equations
      {
          U               0.85;
          k               0.7;
```

```
        epsilon         0.7;
        /*R              0.7;
        nuTilda         0.7; */
    }
}
```

5.4. Fichero de datos para el caso $k\ \omega-$SST a $Re = 10^4$

5.4.1. Capeta 0

5.4.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 6.74e-11;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
type fixedValue;
value uniform 6.74e-11;
    }

    up
```

```
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 6.74e-11;
    }
}


// ************************************************************************* //
```

5.4.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  9.375e-8;

boundaryField
{

    fontAndBack
    {
        type            empty;
```

```
    }

    inlet
    {
        type            fixedValue;
value uniform 9.375e-8;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            kqRWallFunction;
value uniform 9.375e-8;
    }
}


// ************************************************************************* //
```

5.4.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];
```

```
internalField   uniform 0;

boundaryField
{

    fontAndBack
    {
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.4.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
```

```
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

5.4.1.5. Archivo omega

```
/*--------------------------------*- C++ -*----------------------------------*\
```

```
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [ 0 0 -1 0 0 0 0 ];

internalField   uniform 4.86e-2;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 4.86e-2;
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            omegaWallFunction;
        //U              Urel;
        value           uniform 4.86e-2;
    }

   up
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

}
```

5.4.1.6. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
        type            zeroGradient;
    }

    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }

     fontAndBack
     {
type empty;
     }
```

```
}

// ************************************************************************* //
```

5.4.1.7. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.01 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.01 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }

    down
    {
type slip;
```

```
    }
// ************************************************************************* //
```

5.4.2. Carpeta constant

5.4.2.1. Archivo RASProperties

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kOmegaSST;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.4.2.2. Archivo transportProperties

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.4.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //
```

5.4.3. Carpeta polyMesh

5.4.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])
```

```
define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Mathematicalconstants:
define(pi, 3.1415926536)

// Numeros de nodos
define(n1, 5)
define(n2, 21)
define(n3, 139)
define(n4, 139)
define(n5, 289)

// Radios
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))

define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
```

```
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))

// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)

// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 0.5)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


vertices
(
// Arco Derecho
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)

// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)

// Arco Izquierdo
//**************
```

```
// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******

(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)

(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
```

```
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//**************
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//**************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//**************
arc v5b v7b (-Rext 0 z0) // Arco exterior
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)

// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)

// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)
```

```
// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)

// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}

up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}

down
{
type patch;
```

```
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
);
```

5.4.4. Carpeta system

5.4.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam; //boundaryFoam;

startFrom       latestTime;

startTime       24000;

stopAt          endTime;

endTime         26000;

deltaT          2.0;

writeControl    timeStep;

writeInterval   500;

purgeWrite      2;

writeFormat     ascii;
```

```
writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //

functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 1e-2;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.4.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version  2.0;
  format ascii;
```

```
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 12;

method simple;

simpleCoeffs
{
  n (3 4 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();

// ************************************************************************* //
```

5.4.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    //div(phi,epsilon) Gauss limitedLinear 1;
    //div(phi,omega)  bounded Gauss linear;
    div(phi,epsilon) Gauss linear 1;
    div(phi,omega)  Gauss linear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.4.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|omega|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

```
// ************************************************************************* //

relaxationFactors
{
    fields
    {
p 0.3;
    }
    equations
    {
        U               0.7;
        k               0.7;
        epsilon         0.7;
        omega           0.7;
        R               0.7;
        nuTilda         0.7;
    }
}
```

5.5. Fichero de datos para el caso $k - \epsilon$ a $Re = 5 \cdot 10^4$

5.5.1. Capeta 0

5.5.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 8.4227e-09;


boundaryField
{
```

```
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
type fixedValue;
value uniform 8.4227e-09;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 8.4227e-09;
    }
}


// ************************************************************************* //
```

5.5.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
```

```
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  2.3438e-06;

boundaryField
{

    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            fixedValue;
value uniform 2.3438e-06;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            kqRWallFunction;
value uniform 2.3438e-06;
    }
}


// ************************************************************************* //
```

5.5.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    fontAndBack
    {
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
```

```
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.5.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
```

```
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

5.5.1.5. Archivo omega

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [ 0 0 -1 0 0 0 0 ];

internalField   uniform 2.43005e-01;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 2.43005e-01;
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            omegaWallFunction;
        //U               Urel;
        value           uniform 2.43005e-01;
```

```
    }

   up
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

}
```

5.5.1.6. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
        type            zeroGradient;
    }
```

```
    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }

     fontAndBack
     {
type empty;
     }
}

// ************************************************************************* //
```

5.5.1.7. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.05 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.05 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }
```

```
    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }

    down
    {
type slip;
    }
// ************************************************************************* //
```

5.5.2. Carpeta constant

5.5.2.1. Archivo RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kOmegaSST;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.5.2.2. Archivo transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.5.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //
```

5.5.3. Carpeta polyMesh

5.5.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])

define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Mathematicalconstants:
define(pi, 3.1415926536)

// Numeros de nodos
define(n1, 10)
define(n2, 28)
define(n3, 186)
define(n4, 186)
define(n5, 386)

// Radios
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
```

```
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))

define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))

// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)

// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 1)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


vertices
(
// Arco Derecho
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)
```

```
// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)

// Arco Izquierdo
//**************

// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******

(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)
```

```
(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
```

```
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//**************
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//**************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//**************
arc v5b v7b (-Rext 0 z0) // Arco exterior
```

```
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)

// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
```

```
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)

// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)

// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)

// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}
```

```
up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}

down
{
type patch;
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
);
```

5.5.4. Carpeta system

5.5.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam; //boundaryFoam;

startFrom       latestTime;
```

```
startTime       15000;

stopAt          endTime;

endTime         17000;

deltaT          0.20;

writeControl    timeStep;

writeInterval   500;

purgeWrite      2;

writeFormat     ascii;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //

functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 5e-2;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.5.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version  2.0;
  format ascii;
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 16;

method simple;

simpleCoeffs
{
  n (4 4 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();

// ************************************************************************* //
```

5.5.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
```

```
|   \\  /    A nd           | Web:      www.OpenFOAM.org                    |
|    \\/     M anipulation  |                                               |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    //div(phi,epsilon) Gauss limitedLinear 1;
    //div(phi,omega)  bounded Gauss linear;
    div(phi,epsilon) Gauss linear 1;
    div(phi,omega)  Gauss linear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
```

```
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.5.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|omega|R|nuTilda)"
    {
        solver          smoothSolver;
```

```
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //

relaxationFactors
{
    fields
    {
p 0.3;
    }
    equations
    {
        U               0.7;
        k               0.7;
        epsilon         0.7;
        omega           0.7;
        R               0.7;
        nuTilda         0.7;
    }
}
```

5.6. Fichero de datos para el caso $k - \epsilon$ a $Re = 10^5$

5.6.1. Capeta 0

5.6.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

```
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 6.7382e-08;


boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
type fixedValue;
value uniform 6.7382e-08;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 6.7382e-08;
```

```
    }
}


// ************************************************************************* //
```

5.6.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  9.3750e-06;

boundaryField
{

    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            fixedValue;
value uniform 9.3750e-06;
    }

    up
    {
        type            zeroGradient;
    }
```

```
    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            kqRWallFunction;
value uniform 9.3750e-06;
    }
}


// ************************************************************************* //
```

5.6.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
```

```
    fontAndBack
    {
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.6.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
```

```
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}
```

```
// ************************************************************************* //
```

5.6.1.5. Archivo omega

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [ 0 0 -1 0 0 0 0 ];

internalField   uniform 4.86010e-01;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 4.86010e-01;
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            omegaWallFunction;
        //U              Urel;
        value           uniform 4.86010e-01;
    }

   up
    {
```

```
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

}
```

5.6.1.6. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
```

```
        type            zeroGradient;
    }

    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }

     fontAndBack
     {
type empty;
     }
}

// ************************************************************************* //
```

5.6.1.7. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.1 0 0);

boundaryField
{
    inlet
    {
```

```
        type            fixedValue;
        value           uniform (0.1 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }

    down
    {
type slip;
    }
// ************************************************************************* //
```

5.6.2. Carpeta constant

5.6.2.1. Archivo RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
RASModel        kOmegaSST;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.6.2.2. Archivo transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.6.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
```

```
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RASModel;


// ************************************************************************* //
```

5.6.3. Carpeta polyMesh

5.6.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])

define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Mathematicalconstants:
define(pi, 3.1415926536)
```

```
// Numeros de nodos
define(n1, 10)
define(n2, 28)
define(n3, 186)
define(n4, 186)
define(n5, 386)

// Radios
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))

define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))
```

```
// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)

// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 1)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


vertices
(
// Arco Derecho
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)

// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)

// Arco Izquierdo
//**************

// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
```

```
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******

(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)

(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
```

```
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
```

```
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//***************
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//***************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//***************
```

```
arc v5b v7b (-Rext 0 z0) // Arco exterior
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)

// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
```

```
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)

// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)

// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)

// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
```

```
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}

up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}

down
{
type patch;
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
);
```

5.6.4. Carpeta system

5.6.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam; //boundaryFoam;

startFrom       latestTime;

startTime       15000;

stopAt          endTime;

endTime         15100;

deltaT          0.05;

writeControl    timeStep;

writeInterval   200;

purgeWrite      2;

writeFormat     ascii;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //
```

```
functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 1e-1;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.6.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version  2.0;
  format ascii;
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 32;
```

```
method simple;

simpleCoeffs
{
  n (8 4 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
}

distributed no;

roots ();

// ************************************************************************* //
```

5.6.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
```

```
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    //div(phi,epsilon) Gauss limitedLinear 1;
    //div(phi,omega)  bounded Gauss linear;
    div(phi,epsilon) Gauss linear 1;
    div(phi,omega)  Gauss linear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.6.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|omega|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
```

```
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //

relaxationFactors
{
    fields
    {
p 0.3;
    }
    equations
    {
        U               0.7;
        k               0.7;
        epsilon         0.7;
        omega           0.7;
        R               0.7;
        nuTilda         0.7;
    }
}
```

5.7. Fichero de datos para el caso $k - \epsilon$ a $Re = 10^6$

5.7.1. Capeta 0

5.7.1.1. Archivo epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

```
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 6.7382e-05;


boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        /*type            turbulentMixingLengthDissipationRateInlet;
mixingLength 0.01;
phi phi;
k k;*/
type fixedValue;
value uniform 6.7382e-05;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
type epsilonWallFunction;
value uniform 6.7382e-05;
    }
}


// ************************************************************************* //
```

5.7.1.2. Archivo k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform  9.3750e-04;

boundaryField
{

    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            fixedValue;
value uniform 9.3750e-04;
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
```

```
        type            kqRWallFunction;
value uniform 9.3750e-04;
    }
}


// ************************************************************************* //
```

5.7.1.3. Archivo nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    fontAndBack
    {
        type            empty;

    }

    inlet
    {
        type            calculated;
        value           uniform 0;
    }

    up
    {
        type            calculated;
        value           uniform 0;
    }

    outlet
```

```
    {
        type            calculated;
        value           uniform 0;
    }

    down
    {
        type            calculated;
        value           uniform 0;
    }

    hole
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

5.7.1.4. Archivo nuTilda

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nuTilda;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    fontAndBack
    {
        type            empty;
    }

    inlet
    {
        type            zeroGradient;
```

```
    }

    up
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

5.7.1.5. Archivo omega

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [ 0 0 -1 0 0 0 0 ];

internalField   uniform 4.86010e+00;

boundaryField
{
    inlet
    {
        type            fixedValue;
```

```
        value           uniform 4.86010e+00;
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            omegaWallFunction;
        //U              Urel;
        value           uniform 4.86010e+00;
    }

   up
    {
        type            zeroGradient;
    }

    down
    {
        type            zeroGradient;
    }

}
```

5.7.1.6. Archivo p

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
    {
```

```
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    hole
    {
        type            zeroGradient;
    }

    up
    {
type zeroGradient;
    }

    down
    {
type zeroGradient;
    }

     fontAndBack
     {
type empty;
     }
}

// ************************************************************************* //
```

5.7.1.7. Archivo U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (1 0 0);
```

```
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    hole
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    up
    {
type slip;
    }

    down
    {
type slip;
    }
// ************************************************************************* //
```

5.7.2. Carpeta constant

5.7.2.1. Archivo RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kOmegaSST;
```

```
turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

5.7.2.2. Archivo transportProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;

// ************************************************************************* //
```

5.7.2.3. Archivo turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
simulationType  RASModel;


// ************************************************************************* //
```

5.7.3. Carpeta polyMesh

5.7.3.1. Archivo blockMeshDict.m4

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

changecom(//)changequote([,])

define(calc, [esyscmd(perl -e 'use Math::Trig; print ($1)')])

define(VCOUNT, 0)

define(vlabel, [[// ]Vertex $1 = VCOUNT define($1, VCOUNT)define([VCOUNT], incr(VCOUNT))])

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// PARAMETROS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Mathematicalconstants:
define(pi, 3.1415926536)

// Numeros de nodos
define(n1, 23)//inicialmente son 20
define(n2, 21)
define(n3, 139)
define(n4, 139)
define(n5, 289)

// Radios
define(Rint, 0.5)
define(Rext, 1)

// Longitudes del dominio
```

```
define(backLength, calc(20*Rint*2))
define(frontLength, calc(10*Rint*2))
define(highLength, calc(10*Rint*2))

// Origen de coordenadas
define(ox, -backLength)
define(oy, -highLength)

// Angulo
define(alpha, calc(45*pi/180))

// Puntos alrededor de hole
define(PintX1, calc(Rint*cos(-alpha)))
define(PintY1, calc(Rint*sin(-alpha)))

define(PextX1, calc(Rext*cos(-alpha)))
define(PextY1, calc(Rext*sin(-alpha)))

define(PextX2, calc(Rext*cos(alpha)))
define(PextY2, calc(Rext*sin(alpha)))

define(PintX2, calc(Rint*cos(alpha)))
define(PintY2, calc(Rint*sin(alpha)))

define(PextX3, calc(Rext*cos(3*alpha)))
define(PextY3, calc(Rext*sin(3*alpha)))

define(PintX3, calc(Rint*cos(3*alpha)))
define(PintY3, calc(Rint*sin(3*alpha)))

define(PextX4, calc(Rext*cos(5*alpha)))
define(PextY4, calc(Rext*sin(5*alpha)))

define(PintX4, calc(Rint*cos(5*alpha)))
define(PintY4, calc(Rint*sin(5*alpha)))

// Coordenada Z
define(z0, 0.0)
define(z1, 1.0)

// Pendiente y gradiente
define(pend, 0)
define(pend2, calc(0.5*pend))
define(grad, 1)
define(grad2, 20)

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// VERTICES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


vertices
(
// Arco Derecho
```

```
//*************

// Bottom
(PintX1 PintY1 z0) vlabel(v1b)
(PextX1 PextY1 z0) vlabel(v2b)
(PextX2 PextY2 z0) vlabel(v3b)
(PintX2 PintY2 z0) vlabel(v4b)

// Top
(PintX1 PintY1 z1) vlabel(v1t)
(PextX1 PextY1 z1) vlabel(v2t)
(PextX2 PextY2 z1) vlabel(v3t)
(PintX2 PintY2 z1) vlabel(v4t)

// Arco Superior
//**************

// Bottom
(PextX3 PextY3 z0) vlabel(v5b)
(PintX3 PintY3 z0) vlabel(v6b)

// Top
(PextX3 PextY3 z1) vlabel(v5t)
(PintX3 PintY3 z1) vlabel(v6t)

// Arco Izquierdo
//**************

// Bottom
(PextX4 PextY4 z0) vlabel(v7b)
(PintX4 PintY4 z0) vlabel(v8b)

// Top
(PextX4 PextY4 z1) vlabel(v7t)
(PintX4 PintY4 z1) vlabel(v8t)

// Inlet
//******

(-frontLength highLength z0) vlabel(v9b)
(-frontLength calc(PextY3+frontLength*(pend)) z0) vlabel(v10b)
(-frontLength calc(PextY4-frontLength*(pend)) z0) vlabel(v11b)
(-frontLength -highLength z0) vlabel(v12b)

(-frontLength highLength z1) vlabel(v9t)
(-frontLength calc(PextY3+frontLength*(pend)) z1) vlabel(v10t)
(-frontLength calc(PextY4-frontLength*(pend)) z1) vlabel(v11t)
(-frontLength -highLength z1) vlabel(v12t)

// Down
//*****

(calc(PextX4-highLength*pend) -highLength z0) vlabel(v13b)
(calc(PextX1+highLength*pend) -highLength z0) vlabel(v14b)
```

```
(backLength -highLength z0) vlabel(v15b)

(calc(PextX4-highLength*pend) -highLength z1) vlabel(v13t)
(calc(PextX1+highLength*pend) -highLength z1) vlabel(v14t)
(backLength -highLength z1) vlabel(v15t)

// Outlet
//*******

(backLength calc(PextY1-backLength*pend2) z0) vlabel(v16b)
(backLength calc(PextY2+backLength*pend2) z0) vlabel(v17b)
(backLength highLength z0) vlabel(v18b)

(backLength calc(PextY1-backLength*pend2) z1) vlabel(v16t)
(backLength calc(PextY2+backLength*pend2) z1) vlabel(v17t)
(backLength highLength z1) vlabel(v18t)

// Up
//***

(calc(PextX2+highLength*pend) highLength z0) vlabel(v19b)
(calc(PextX3-highLength*pend) highLength z0) vlabel(v20b)

(calc(PextX2+highLength*pend) highLength z1) vlabel(v19t)
(calc(PextX3-highLength*pend) highLength z1) vlabel(v20t)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// BLOQUES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

blocks
(
// Arco derecho
hex (v1b v2b v3b v4b v1t v2t v3t v4t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco superior
hex (v4b v3b v5b v6b v4t v3t v5t v6t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco izquierdo
hex (v6b v5b v7b v8b v6t v5t v7t v8t)
(n1 n2 1)
simpleGrading (grad2 1 1)

// Arco inferior
hex (v8b v7b v2b v1b v8t v7t v2t v1t)
(n1 n2 1)
simpleGrading (grad2 1 1)
```

```
// Bloque 9
hex (v5b v10b v11b v7b v5t v10t v11t v7t)
(n3 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 10
hex (v7b v11b v12b v13b v7t v11t v12t v13t)
(n3 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 11
hex (v7b v13b v14b v2b v7t v13t v14t v2t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 12
hex (v2b v14b v15b v16b v2t v14t v15t v16t)
(n4 n5 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 5
hex (v2b v16b v17b v3b v2t v16t v17t v3t)
(n5 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 6
hex (v3b v17b v18b v19b v3t v17t v18t v19t)
(n5 n4 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

// Bloque 7
hex (v3b v19b v20b v5b v3t v19t v20t v5t)
(n4 n2 1)
edgeGrading (grad grad grad grad  1 1 1 1  1 1 1 1)

// Bloque 8
hex (v5b v20b v9b v10b v5t v20t v9t v10t)
(n4 n3 1)
edgeGrading (grad 1 1 grad  grad 1 1 grad  1 1 1 1)

);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// EJES
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

edges
(
// Arco Derecho
//**************
arc v2b v3b (Rext 0 z0) // Arco exterior
arc v4b v1b (Rint 0 z0) // Arco interior

arc v2t v3t (Rext 0 z1) // Arco exterior
```

```
arc v4t v1t (Rint 0 z1) // Arco interior

// Arco Superior
//**************
arc v3b v5b (0 Rext z0) // Arco exterior
arc v6b v4b (0 Rint z0) // Arco interior

arc v3t v5t (0 Rext z1) // Arco exterior
arc v6t v4t (0 Rint z1) // Arco interior

// Arco Izquierdo
//**************
arc v5b v7b (-Rext 0 z0) // Arco exterior
arc v8b v6b (-Rint 0 z0) // Arco interior

arc v5t v7t (-Rext 0 z1) // Arco exterior
arc v8t v6t (-Rint 0 z1) // Arco interior

// Arco Inferior
//**************
arc v7b v2b (0 -Rext z0) // Arco exterior
arc v1b v8b (0 -Rint z0) // Arco interior

arc v7t v2t (0 -Rext z1) // Arco exterior
arc v1t v8t (0 -Rint z1) // Arco interior
);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
// CONTORNOS
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

boundary
(
hole
{
type wall;
faces
(
(v1b v4b v4t v1t) // Arco derecho
(v4t v6t v6b v4b) // Arco superior
(v6b v8b v8t v6t) // Arco izquierdo
(v1b v8b v8t v1t) // Arco inferior
);
}

frontAndBack
{
type empty;
faces
(
// Arco derecho
(v1b v2b v3b v4b)
(v1t v2t v3t v4t)
```

```
// Arco superior
(v4t v3t v5t v6t)
(v4b v3b v5b v6b)

// Arco izquierdo
(v6b v5b v7b v8b)
(v6t v5t v7t v8t)

// Arco inferior
(v8b v7b v2b v1b)
(v8t v7t v2t v1t)

// Bloque 5
(v3b v2b v16b v17b)
(v3t v2t v16t v17t)

// Bloque 6
(v18b v19b v3b v17b)
(v18t v19t v3t v17t)

// Bloque 7
(v19b v20b v5b v3b)
(v19t v20t v5t v3t)

// Bloque 8
(v20b v9b v10b v5b)
(v20t v9t v10t v5t)

// Bloque 9
(v10b v11b v7b v5b)
(v10t v11t v7t v5t)

// Bloque 10
(v11b v12b v13b v7b)
(v11t v12t v13t v7t)

// Bloque 11
(v7b v13b v14b v2b)
(v7t v13t v14t v2t)

// Bloque 12
(v2b v14b v15b v16b)
(v2t v14t v15t v16t)
);
}

inlet
{
type patch;
faces
(
(v9b v10b v10t v9t)
(v10b v11b v11t v10t)
(v11b v12b v12t v11t)
```

```
);
}

outlet
{
type patch;
faces
(
(v18b v17b v17t v18t)
(v17b v16b v16t v17t)
(v16b v15b v15t v16t)
);
}

up
{
type patch;
faces
(
(v20b v9b v9t v20t)
(v19b v20b v20t v19t)
(v18b v19b v19t v18t)
);
}

down
{
type patch;
faces
(
(v12b v13b v13t v12t)
(v13b v14b v14t v13t)
(v14b v15b v15t v14t)
);
}
);

mergePatchPairs
(
);
```

5.7.4. Carpeta system

5.7.4.1. Archivo controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam; //boundaryFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         3000;

deltaT          0.010;

writeControl    timeStep;

writeInterval   100;

purgeWrite      2;

writeFormat     ascii;

writePrecision  6;

writeCompression on;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;


// ************************************************************************* //

functions
{
forces
{
type forceCoeffs;
functionObjectLibs ( "libforces.so" );
outputControl timeStep;
outputInterval 1;
patches ( "hole" );
pName p;
UName U;
rhoName rhoInf;
```

```
log true;
rhoInf 1000;
liftDir (0 1 0);
dragDir (1 0 0);
pitchAxis (0 0 1);
CofR (0 0 0);
magUInf 1;
lRef 1;
Aref 1;
}
}

// ************************************************************************* //
```

5.7.4.2. Archivo decomposePartDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
  version  2.0;
  format ascii;
  class dictionary;
  location "system";
  object decomposePartDict;
}


numberOfSubdomains 16;

method simple;

simpleCoeffs
{
  n (4 4 1);
  delta 0.001;
}

hierarchicalCoeffs
{
  n (1 1 1);
  delta 0.001;
  order xyz;
}

manualCoeffs
{
  dataFila "";
```

```
}

distributed no;

roots ();

// ************************************************************************* //
```

5.7.4.3. Archivo fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    //div(phi,epsilon) Gauss limitedLinear 1;
    //div(phi,omega)  bounded Gauss linear;
    div(phi,epsilon) Gauss linear 1;
    div(phi,omega)  Gauss linear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
```

```
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

5.7.4.4. Archivo fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
```

```
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|epsilon|omega|R|nuTilda)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //

relaxationFactors
{
    fields
    {
p 0.3;
    }
    equations
    {
        U               0.7;
        k               0.7;
        epsilon         0.7;
        omega           0.7;
        R               0.7;
        nuTilda         0.7;
    }
}
```